Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

META-MODELO FUNCIONAL PARA RECUPERAÇÃO DE INFORMAÇÃO BASEADO EM λ-CÁLCULO

Daniel Gonzaga dos Santos

Uberlândia - MG
Dezembro de 2006

META-MODELO FUNCIONAL PARA RECUPERAÇÃO DE INFORMAÇÃO BASEADO EM λ-CÁLCULO

Por

Daniel Gonzaga dos Santos

DISSERTAÇÃO APRESENTADA À UNIVERSIDADE FEDERAL DE UBERLÂNDIA, MINAS GERAIS, COMO PARTE DOS REQUISITOS EXIGIDOS PARA OBTENÇÃO DO TÍTULO DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO

Área de concentração: Banco de Dados.

Orientador: João Nunes de Souza - UFU
Co-Orientrador: Ilmério Reis e Silva - UFU

DEZEMBRO DE 2006

FICHA CATALOGRÁFICA
Elaborado pelo Sistema de Bibliotecas da UFU / Setor de Catalogação e Classificação

S237m	Santos, Daniel Gonzaga dos, 1982- Meta-Modelo Funcional para Recuperação de Informação baseado em λ-cálculo / Daniel Gonzaga dos Santos. - Uberlândia, 2006. 100f. : il. Orientador: João Nunes de Souza. Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação. Inclui bibliografia. 1. Recuperação da informação - Teses. 2. Banco de Dados - Teses. I. Souza, João Nunes de. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título. CDU: 6813.07

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada "**Meta-Modelo Funcional para Recuperação de Informação baseado em λ-cálculo**" por **Daniel Gonzaga dos Santos** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 18 de dezembro de 2006

Banca Examinadora:

Prof. Dr. João Nunes de Souza - Orientador
Universidade Federal de Uberlândia UFU / MG

Prof. Dr. Marcos André Gonçalves
Universidade Federal de Minas Gerais UFMG / MG

Prof. Dr. Marcelo de Almeida Maia
Universidade Federal de Uberlândia UFU / MG

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Data: Dezembro, 2006

Autor:	**Daniel Gonzaga dos Santos**
Título:	**Meta-Modelo Funcional para Recuperação de Informação baseado em λ-cálculo**
Faculdade:	**Faculdade de Computação**
Grau:	**Mestrado**

_____

Autor

Dedicatória

Aos meus pais Silvano e Walquíria, aos meus irmãos Breno e Ana Laura e a minha namorada Miriã

Agradecimentos

Primeiramente, agradeço a Deus, porque Dele, por Ele e para Ele são todas as coisas. Glorifico a Deus pois Ele é a minha fonte de sabedoria e meu sustento.

Ao meu orientador e ao meu co-orientador, Profs. Drs. João N. Souza e Ilmério R. Silva, sou profundamente grato pela orientação, pelos conselhos, pela paciência, por suas contribuições e acima de tudo pela amizade e pela confiança depositada em mim durante o desenvolvimento deste trabalho.

Aos membros da banca, Prof. Dr. Marcos André Gonçalves e Prof. Dr. Marcelo de Almeida Maia pelas colaborações e contribuições.

À toda minha família, pelo incentivo, apoio e amor durante este mestrado, especialmente aos meus queridos pais Silvano e Walquíria, aos meus amados irmãos Breno e Ana Laura, ao meu cunhado Leonardo, e a minha namorada Miriã pela compreensão, amor e carinho incondicional. Obrigado por estarem ao meu lado em todos os momentos e por me darem forças para enfrentar os desafios da vida.

À todos os professores e amigos da Pós Graduação que, durante o meu mestrado, compartilharam comigo seus conhecimentos, sem os quais teria sido impossível realizar esse trabalho. Agradeço a todos os colegas da Pós Graduação que estiveram presentes em alguns momentos importantes na elaboração e apresentação deste trabalho, valeu pela amizade, pelos conselhos, pelos trabalhos e estudos que realizamos juntos.

Aos amigos da Igreja Cristã Luz do Mundo, a toda família da minha namorada e a todos aqueles que me apoiaram e oraram por mim durante esta jornada, sou grato e glorifico a Deus pela vida de vocês.

Finalmente, agradeço a CAPES pelo apoio financeiro e a todos que contribuíram de alguma forma para a realização deste trabalho.

Resumo

Modelagem é um tópico de pesquisa central em Recuperação de Informação (RI). Uma abordagem para estudo e desenvolvimento de novos modelos de RI são os *frameworks* genéricos. Estes *frameworks* podem ser vistos como meta-modelos formais que incluem uma notação utilizada para descrever modelos de RI e possibilitar a investigação da semântica do processo de recuperação. Além disso, estes meta-modelos facilitam o raciocínio sobre as características e propriedades de modelos de recuperação de informação.

Nesta dissertação, propomos um *framework* funcional genérico e formal, baseado em λ-cálculo, para definição e estudo de modelos de RI, denominado Estrutura Funcional. Este *framework* (ou meta-modelo) permite a representação, combinação, formulação e comparação de equivalência entre modelos de RI. Neste meta-modelo, os modelos de RI são representados através de funções, ou seja, definimos formalmente os componentes tais como documentos, consultas e função de similaridade de modelos de recuperação de informação, utilizando o λ-cálculo. Esta estratégia tem como elemento principal o conceito de funções, neste sentido difere dos modelos tradicionais, que geralmente consideram pesos de termos, vetores, etc como fundamentos. Ao representarmos os modelos de RI em uma mesma linguagem funcional, como o λ-cálculo, a identificação dos argumentos e valores das funções ficam claros.

Além disso, mostramos exemplos de como representar os modelos clássicos de RI, estudando a equivalência destes com modelos vetoriais alternativos definidos aqui, por meio da Estrutura Funcional. Também representamos nesta estrutura os modelos: redes de crença, *sTerm* e um baseado em ontologia. Com isso, verificamos que a passagem dos modelos de RI para a Estrutura Funcional possibilita a construção e a combinação de modelos, bem como permite compará-los quanto a sua similaridade (equivalência ou não), através de demonstrações algébricas, sem realizar experimentos.

Palavras-chave: Estrutura Funcional, λ-cálculo, Recuperação de Informação, Meta-Modelo.

Abstract

Modeling is a topic of the central research in Information Retrieval (IR). Generic frameworks are approaches for study and development of new IR models is generic frameworks. These frameworks could be seen as formal metamodels that include a notation used to describe IR models and to make possible the research on the semantics retrieval process. Besides that, these metamodels make easier the reasoning on the characteristics and properties of information Retrieval models.

In this dissertation, we offer a generic and formal functional framework, based on λ-calculus, for definition and study of the IR models, also called Functional Structure. This framework (or metamodel) allows the representation, combination formularization and equivalence comparison among IR models. In this metamodel, the IR models are represented by functions, that is, we have formally defined the components such as documents, consultations and similarity function of information Retrieval models, using the λ-calculus. This strategy has as the main element the functions concept, in this meaning it differs from the traditional models, which generally consider weights of terms, vectors, etc as beddings. When representing the IR models in a same functional language, as the λ-calculus, the identification of the arguments and values of the functions are clear.

Moreover, we show examples of how to represent the classic IR models, studying the equivalence of alternative vectorial models, defined here using the Functional Structure. We also represent in this structure the following models: belief networks, sTerm and a based on ontology. By this, we verify that the representation of the IR models to the Functional Structure makes possible the construction and the combination of models, it also allows the comparison among them by its similarity (equivalence or not), through algebraic demonstrations, without carrying through experiments.

Keywords: Functional structure, λ-calculus, Information Retrieval, Formal metamodel

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

FNB - Forma Normal de *Backus*

FND - Forma Normal Disjuntiva

OWL - *Web Ontology Language*

RDL - Raciocinador de Descrição Lógica

RI - Recuperação de Informação

SRI - Sistema de Recuperação de Informação

URL - Universal Resource Locator

XML - *eXtensible Markup Language*

Lista de Símbolos

λ	-	Conectivo utilizado para definir funções
d_j	-	j-ésimo documento da coleção
k_i	-	i-ésimo termo de um documento
q	-	Consulta do usuário
$w_{i,j}$	-	Função peso do termo k_i no documento d_j
$w_{i,q}$	-	Função peso do termo k_i no documento q
tf	-	Termo funcional
$\mathbf{K}$	-	Conjunto de termos da coleção
$L_{\mathbf{K}}$	-	Conjunto de todas as listas formadas com elementos de $\mathbf{K}$
$peso_j$	-	Termo funcional que define o peso de um termo no documento d_j
$peso_q$	-	Termo funcional que define o peso de um termo no documento q
df_j	-	j-ésimo documento funcional de uma coleção na estrutura funcional
qf	-	Consulta funcional do usuário na estrutura funcional
$\mathbf{U}_f$	-	Universo funcional
$\mathbf{D}_f$	-	A lista de documentos funcionais
$\mathbf{Q}_f$	-	A lista de consultas funcionais
$\mathbf{C}_f$	-	Coleção de referência funcional
Ψ	-	Modelo funcional
$\Delta(of_j, of_i)$	-	Função similaridade entre dois objetos funcionais of_j e of_i na estrutura funcional
Δ_n	-	Função similaridade que satisfaz a normalização
Δ_r	-	Função similaridade que satisfaz a reflexividade
Δ_s	-	Função similaridade que satisfaz a simetria
Δ_{nrs}	-	Função similaridade que satisfaz a normalização, a reflexividade e simetria
Ψ^v	-	Modelo funcional vetorial
N	-	Número de documentos na coleção
n_i	-	Número de documentos no qual o termo k_i aparece
$freq_{i,j}$	-	Frequência natural do termo k_i no documento d_j, isto é, número de vezes que k_i aparece em d_j
idf_i	-	Freqüência inversa de documentos do termo k_i em uma coleção
$sim(d_j, q)$	-	Função similaridade entre o documento d_j e a consulta q

Ψ^p	-	Modelo funcional probabilístico
V_p	-	Modelo vetorial-probabilístico
Δ^p	-	Função similaridade do modelo probabilístico na estrutura funcional
Ψ^{vp}	-	Modelo funcional vetorial-probabilístico
Δ^{vp}	-	Função similaridade do modelo vetorial-probabilístico na estrutura funcional
$\simeq$	-	Comparar modelos funcionais quanto a sua equivalência
$\mathbf{R}_q$	-	Conjunto de documentos relevantes para a consulta q
$\overline{\mathbf{R}}_q$	-	O complemento de $\mathbf{R}_q$
$P(k_i\|\mathbf{R}_q)$	-	Probabilidade de um termo k_i estar presente em um documento escolhido aleatoriamente do conjunto $\mathbf{R}_q$
$P(\mathbf{R}_q\|d_j)$	-	Probabilidade do documento d_j ser relevante para a consulta q no conjunto $\mathbf{R}$
$R_{j,q}$	-	Função de *ranking* ou similaridade calculada pelo modelo vetorial
$prob_q$	-	Termo funcional que representa a probabilidade de um termo k_i estar presente em um documento escolhido aleatoriamente do conjunto $\mathbf{R}_q$
$\overline{prob}_q$	-	Termo funcional que representa a probabilidade de um termo k_i estar presente em um documento escolhido aleatoriamente do conjunto $\overline{\mathbf{R}}_q$
δ_q	-	Termo funcional que serve como um fator discriminante para a consulta q
Ψ^b	-	Modelo funcional booleano
Δ^b	-	Função similaridade do modelo booleano na estrutura funcional
V_b	-	Modelo vetorial-booleano
Ψ^{vb}	-	Modelo funcional vetorial-booleano
Δ^{vb}	-	Função similaridade do modelo vetorial-booleano na estrutura funcional
bol_j	-	Termo funcional que representa a função booleana da conjunção de termos pertencentes ao documento d_j e a negação dos termos que não pertencem ao documento d_j
bol_q	-	Termo funcional que retorna uma expressão booleana associada a consulta q
Ψ^{rc}	-	Modelo funcional redes-crença
Δ^{rc}	-	Função similaridade do modelo redes-crença na estrutura funcional
V_{rc}	-	Modelo vetorial-redes-crença
Ψ^{vrc}	-	Modelo funcional vetorial-redes-crença
Δ^{rc}	-	Função similaridade do modelo vetorial redes-crença na estrutura funcional
$peso_{ei_j}$	-	Termo funcional que define um valor para a evidência ei associada ao documento d_j
$E_{i,j}$	-	Valor da i-ésima evidência em relação ao documento d_j
$E_{i,q}$	-	Valor da i-ésima evidência em relação a consulta q
Ψ^{ows}	-	Modelo funcional do modelo baseado em ontologia para web semântica.
Δ^{ows}	-	Função similaridade do modelo baseado em ontologia na estrutura funcional
$pesoSem_j$	-	Termo funcional que define o peso de um termo semântico no documento d_j
Ψ^{st}	-	Modelo funcional do modelo *sTerm*
Δ^{st}	-	Função similaridade do modelo *sTerm* na estrutura funcional
V_{st}	-	Modelo vetorial-estrutural
Ψ^{vst}	-	Modelo funcional vetorial-estrutural
Δ^{vst}	-	Função similaridade do modelo vetorial-estrutural na estrutura funcional
$peso_j^{st}$	-	Termo funcional que define o peso de um termo estrutural no documento d_j
$tf_{T,D}$	-	Frequência do termo estrutural T no documento D.
idf_T^t	-	Frequência inversa do termo estrutural T no documento de tipo t
D^t	-	Documento do tipo t

Capítulo 1

Introdução

A área de Recuperação de Informação (RI) possui grande importância em Ciência da Computação há várias décadas e tem experimentado um maior interesse da comunidade científica devido à grande disponibilidade de documentos existentes hoje na forma digital, principalmente na *Web*. Recuperação de informação estuda a representação, armazenamento, organização e a recuperação automática de documentos [3]. Atualmente, as pesquisas em RI incluem: modelagem, categorização e classificação de documentos, arquitetura de sistemas, interface do usuário, etc. Dentre estas, a modelagem é um dos tópicos de pesquisa centrais e ativos em RI. Neste trabalho propomos um meta-modelo baseado em λ-cálculo, chamado Estrutura Funcional, como uma ferramenta para ajudar projetistas na tarefa de desenvolvimento, representação e comparação de modelos de RI. Neste capítulo, descrevemos alguns conceitos de recuperação de informação, apresentamos os principais trabalhos relacionados a nossa proposta, discutimos os objetivos e as contribuições de nosso trabalho.

1.1 Recuperação de Informação

Recuperação de Informação é uma área que estuda o armazenamento, classificação, agrupamento e recuperação automática de documentos. A abundância de informações na *Web* é uma das principais razões para sua crescente popularidade. A facilidade do uso e acessibilidade da *Web* fez dela uma ferramenta muito importante, não somente para comunicação, mas também para armazenamento e compartilhamento de informação. Do ponto de vista de RI, a *Web* pode ser vista como grande repositório

de dados contendo documentos, ou páginas *Web*, que são interconectados.

O problema central em recuperação de informação é encontrar informações de interesse dos usuários e a principal ferramenta usada para resolver este problema é o emprego de sistemas de recuperação de informação (SRI). A área de recuperação de informação apresentou importantes resultados, desde o seu início, nas tarefas de localizar e classificar documentos em sistemas bibliográficos, até então restritos a bibliotecas e redes de menor escala. Mais recentemente, o emprego de RI para busca de informações na *Web* contribuiu enormemente para a criação de máquinas de buscas.

O usuário de um SRI busca obter documentos que atendam à sua necessidade de informação. Para isto, ele, geralmente, expressa sua necessidade por meio de uma consulta. O sistema então realiza o casamento da consulta com seus documentos por meio de um modelo de recuperação de informação. Um modelo para RI propõe uma representação para documentos e consultas e define como calcular a similaridade entre os mesmos. O modelo, então, é capaz de quantificar a relevância de cada documento da coleção em relação à consulta do usuário e, com base nesta relevância, apresentar-lhe os documentos que melhor atendam à sua necessidade de informação i.e., os mais similares à consulta. Por fim, o modelo apresenta um *ranking* de documentos que aparecem em ordem decrescente de similaridade. Para realizar a tarefa de recuperar documentos, um SRI trabalha com modelos de representação de documentos e consultas. Diversos modelos de recuperação de informação têm sido estudados e propostos em RI. Dentre eles estão os modelos de espaço vetorial [3, 22, 46], modelos probabilísticos [24, 34, 41, 50] e modelos baseados em lógica [9, 23, 52]. Os tipos de modelos de RI desenvolvidos podem ser divididos em: modelos clássicos como booleano, vetorial e probabilístico [3]; modelos alternativos: por exemplo, redes de crença, vetorial generalizado, booleano generalizado, entre outros [3]; modelos estruturados: como o modelo *proximal nodes*, modelo *sTerm* [3, 47]; modelos lógicos de RI [12, 13]. Existem ainda, modelos baseados em Inteligência Artificial, como modelos baseados em conhecimento, em redes neurais artificiais, em algoritmos genéticos e em linguagem de processamento natural [17, 18, 31].

Os diferentes tipos de modelos de RI refletem a complexidade de RI em geral e, em especial da tarefa de modelagem. Os tipos de modelos de RI diferem, em sua maioria, na forma que os objetos (documentos, imagens,etc) são representados e como a recuperação é definida. Os modelos algébricos ou matemáticos geralmente representam os objetos como uma seqüência de números (tradicional-

mente chamados de vetores), e define a recuperação como um relacionamento entre os números. A Lógica em RI assume os objetos como representações (exemplo, coleções de sentenças) e a recuperação como uma inferência lógica. Os modelos estruturados combinam informação de conteúdo com informação de estrutura dos documentos e representam seus objetos através de termos estruturais. A Inteligência Artificial para RI visualiza os objetos como conhecimento e a recuperação como alguma razão ou como neurônios ou como regiões de ativação.

O presente trabalho apresenta uma representação comum para diversos modelos de RI por meio da proposta de um *framework* genérico baseado em λ-cálculo.

1.2 Trabalhos relacionados

Uma ferramenta utilizada para desenvolvimento de um novo modelo de recuperação de informação são os *frameworks* genéricos que podem ser vistos como meta-modelos formais. Podemos definir um meta-modelo como sendo um arcabouço para representação de modelos de RI. Um meta-modelo ajuda desenvolvedores na produção de novos modelos de recuperação de informação e também pode ser utilizado para representação de modelos anteriormente propostos facilitando o estudo de propriedades e características dos modelos de RI. Além disso, os *frameworks* podem ajudar os desenvolvedores a modificar modelos existentes ou estendê-los de forma que eles se tornem mais eficientes e flexíveis.

Um formalismo em RI geralmente utiliza notação matemática para a representação de estratégias de RI ou modelos de RI. Um meta-modelo formal consiste na notação utilizada para descrever modelos de RI, possibilitando um estudo de suas propriedades e características [19, 37]. Neste contexto, a representação em notação matemática de estratégias de recuperação é um importante assunto de pesquisa em RI. Os meta-modelos permitem que diferentes características dos modelos possam ser combinadas em um mesmo plano de representação.

Os meta-modelos formais na literatura podem ser classificadas como lógicos ou algébricos [37]. Aqui, classificamos os meta-modelos formais em: meta-modelos algébricos, meta-modelos baseados em probabilidades e meta-modelos baseados em lógica. Os meta-modelos baseados em probabilidades são um tipo de meta-modelo algébrico. Observamos que o meta-modelo proposto neste trabalho

pode ser classificado como algébrico.

Meta-Modelos Algébricos. Vários meta-modelos formais algébricos são propostos por Dominich em [19, 20, 21]. Este autor tem realizado um extenso trabalho na formalização de modelos. Em [19, 20], propõe um *framework* definindo alguns conceitos para modelagem de qualquer modelo clássico de RI. Esses trabalhos definem conjunto recuperação, mas não definem *ranking*. O artigo [21] mostra uma definição formal de RI através da medida de uma relação entre documentos e um modelo de usuário, mas não apresenta aplicações práticas para modelos de RI. Em nosso trabalho, mostramos aplicações da estrutura funcional, tais como: representação e comparação de modelos.

Em [39], é proposto um meta-modelo funcional para representação e comparação de modelos, entretanto não possui uma notação precisa para definição das funções. Em nosso trabalho, utilizamos a notação λ-cálculo para definir as funções que caracterizam os modelos. Além disso, foram feitas demonstrações algébricas na comparação entre modelos.

Meta-Modelos Baseados em Probabilidades. Existem alguns trabalhos sobre meta-modelos formais ou *frameworks* genéricos baseados em probabilidades para modelos de RI. Estes *frameworks* são baseados principalmente em redes bayesianas.

As redes bayesianas, introduzida em [40], fornecem um formalismo gráfico para representar independências entre as variáveis de distribuição de probabilidade conjunta.

Turtle e Croft [50] propõe o primeiro modelo de rede bayesiana para RI, onde demonstram que ao estender o modelo básico de rede de inferência com representações booleanas das consultas de usuário poderia se obter um bom desempenho na qualidade do *ranking*.

Um segundo modelo foi proposto por Ribeiro-Neto e Muntz [42], denominado modelo de redes de crença para RI, derivado de considerações probabilísticas. Nesta proposta, a aplicação do modelo de redes de crença para modelos de RI é realizada.

Os documentos, termos e consultas são representados no modelo de crença por variáveis aleatórias binárias e o cálculo do grau de relevância é baseado em probabilidades. Além disso, para simplificar a modelagem, na rede de crença os termos são considerados independentes entre si.

No modelo de redes de crença [42], a consulta e o documento são modelados do mesmo modo

para facilitar a definição da estrutura da rede.

Em [48] é proposto um modelo que representa os três modelos clássicos de RI (vetorial, booleano e probabilístico), os ciclos de realimentação de documentos relevantes e alternativas de similaridade consulta-consulta. Outros modelos de RI existentes também podem ser representados através do modelo de redes de crença. Neste trabalho, representamos no meta-modelo proposto, além dos modelos clássicos, o modelo de redes de crença para combinar múltiplas fontes de evidências.

Meta-Modelos Baseados em Lógica. Alguns trabalhos usam Lógica para definir um meta-modelo para modelos de RI. O artigo [9] mostra um resumo de como pesquisas passadas têm combinado o uso de Lógica e incertezas para formulação de modelos de RI.

O uso da Lógica em RI fornece a capacidade para formulação de modelos genéricos e torna possível o estudo de propriedades desses modelos. Meta-modelos lógicos para RI são estudados para fornecer uma rica e uniforme representação da informação e sua semântica. Geralmente, em um modelo lógico as consultas e documentos podem ser representados por fórmulas lógicas. A inferência é associada com implicação lógica: um documento é relevante para a consulta significa que o documento implica na consulta. Somente a Lógica não é capaz de representar um modelo de RI, a teoria da incerteza é necessária [9].

Existem outros estudos com abordagens lógicas sobre meta-modelos em RI. O uso da Lógica para formalmente conduzir provas para RI foi proposto inicialmente em [38]. Na década passada, diversos meta-modelos têm sido propostos [8, 27]. Meta-modelos lógicos podem ser classificados em três tipos: baseados na teoria da situação, baseados em lógica modal, e outros tipos [35].

1.3 Objetivos e Contribuições

Este trabalho apresenta um meta-modelo funcional baseado em λ-cálculo, denominado estrutura funcional, cujo objetivo é ajudar desenvolvedores na tarefa de desenvolvimento de modelos de RI, permitindo a representação, combinação, construção e comparação de modelos de RI.

Uma caracterização formal dos modelos de RI é apresentado em [3]. Nesta caracterização, chamada aqui por Caracterização BR-Formal, são definidos quatro componentes que um modelo

deve ter: visão lógica de documentos, visão lógica de consultas, um *framework* para modelagem de documentos, consultas e seus relacionamentos, e uma função de *ranking*. Este modelo é completo e rico, mas é muito geral e por isso não é utilizado na prática. Já o meta-modelo proposto neste trabalho, funcional, também define os componentes que um modelo de RI deve possuir e é uma ferramenta para comparar equivalência entre modelos. Também é rica e completa, mas com um nível de abstração menor que o meta-modelo apresentado em [3]. Por exemplo, nosso meta-modelo define as propriedades para a função de similaridade ou *ranking*.

O artigo [37] faz uma revisão de alguns métodos formais para sistemas de RI e propõe um novo meta-modelo formal generalizando as definições de documentos, consultas, função de *ranking* e conjunto recuperação. Este meta-modelo permite a representação dos modelos clássicos. Em nosso meta-modelo, as definições de documentos, consultas, função de *ranking* (similaridade), e conjunto recuperação são definidos. Além disso, todos os modelos de RI expressos por um algoritmo podem ser representados no meta-modelo funcional. Outra vantagem é que nosso trabalho define uma formalização, baseado na notação λ-cálculo, para comparação de equivalência entre modelos.

As principais contribuições desta dissertação são:

- A proposta de um meta-modelo funcional com os seguintes objetivos:
  - utilizar uma notação funcional capaz de fornecer expressividade, capacidade de abstração e identificação de componentes e relações relevantes em problemas de RI;
  - utilizar a notação funcional λ-cálculo para definir e caracterizar as funções presentes nos modelos;
  - ter como elemento central da representação o conceito de funções;
  - ser capaz de representar além dos modelos clássicos, modelos estruturados, modelos que combinam evidências, modelos baseados em ontologia e o conjunto de modelos que podem ser expressos por meio de algoritmos, utilizando a notação λ-cálculo;
  - comparar modelos quanto a sua similaridade (equivalência) ou não, através de demonstrações algébricas, sem realizar experimentos;
  - classificar cada modelo de acordo com características das funções de similaridade, a saber, reflexividade, simetria e normalização.

- A construção de modelos vetoriais alternativos de RI utilizando a estrutura funcional:
  - modelo vetorial-probabilístico;
  - modelo vetorial-booleano;
  - modelo vetorial-estrutural;

Nosso objetivo é apresentar a estrutura funcional baseada em λ-cálculo como um meta-modelo para RI e suas aplicações. Dentre as aplicações do meta-modelo funcional temos a construção de modelos de RI, a comparação sem a necessidade de realizar experimentos e a combinação de modelos de RI. Com este meta-modelo, modelos de RI podem ser representados em uma linguagem comum, a linguagem λ-cálculo, tornando mais fácil o estudo de características e propriedades dos modelos, a combinação e a comparação relativa entre modelos. Assim, através da linguagem funcional, podemos pensar nos modelos de RI em um nível mais alto de abstração.

Tendo em vista que o meta-modelo funcional proposto em [39] é baseado em funções e que a notação λ-cálculo é uma definição para funções, uma das principais motivações de nosso trabalho é a utilização desta notação para formalização do meta-modelo funcional. Este meta-modelo, por ser baseado em funções, se diferencia dos outros meta-modelos propostos na literatura, a saber, baseados em lógica, baseados em probabilidade e outros meta-modelos algébricos. A estrutura funcional não é tão limitada quanto os meta-modelos probabilísticos que são difíceis de serem aplicados em alguns contextos e nem tão abstrata quanto os meta-modelos lógicos que carecem de exemplos de aplicação. O meta-modelo funcional proposto, baseado em λ-cálculo, é prático no sentido de implementação dos problemas de RI e também permite trabalhar com aplicações teóricas.

1.4 Organização da Dissertação

O conteúdo desta dissertação está organizado em 6 capítulos, como descrito a seguir.

No Capítulo 2 introduzimos os fundamentos matemáticos para a proposta de uma estrutura funcional baseada em λ-cálculo. Para tal, definimos e descrevemos alguns dos principais conceitos do λ-cálculo.

No Capítulo 3 formalizamos os conceitos da estrutura funcional para RI utilizando a notação λ-cálculo.

No Capítulo 4 apresentamos os modelos clássicos, representamos estes modelos na estrutura funcional, construímos novos modelos e comparamos os modelos de RI utilizando a estrutura funcional.

No Capítulo 5 apresentamos os modelos: *sTerm*, um modelo baseado em ontologia para *Web* semântica e o modelo de redes de crença para combinar múltiplas fontes de evidências. Além disso, representamos e comparamos estes modelos utilizando a estrutura funcional.

Finalmente, no Capítulo 6 concluímos o trabalho, discutimos as vantagens da utilização da estrutura funcional e apresentamos algumas direções para pesquisas futuras.

Capítulo 2

Fundamentos Matemáticos

Este capítulo introduz os fundamentos matemáticos para a proposta de uma estrutura funcional para RI baseada em λ-cálculo.

2.1 λ-cálculo

O λ-cálculo é um sistema formal que lida com a teoria de funções. Foi introduzido na década de 30 por Alonzo Church com o intuito de capturar os aspectos mais básicos da maneira pela qual operadores ou funções podem ser combinados para formar outros operadores [14].

Tal formalismo serve como uma ponte entre linguagens funcionais de alto nível e suas implementações de baixo nível. Pode-se evidenciar o λ-cálculo como uma linguagem intermediária, extremamente simples, consistindo de poucas construções sintáticas e de uma semântica simples e expressiva [25].

Desta forma pode ser considerado como uma linguagem de programação abstrata, onde o conceito de computação, isto é, as maneiras como funções podem ser combinadas para formar outras funções, aparece em toda sua generalidade e de uma forma "pura", despida de complicações sintáticas. Trata-se de uma linguagem expressiva, a qual é suficientemente poderosa para expressar todos os programas funcionais e, por conseguinte, todas as funções computáveis [26].

As principais vantagens e características da notação em λ-cálculo são as seguintes:

- Simplicidade: só existem três tipos de expressões;

- Completude: todas as funções computáveis podem ser representadas;
- Generalidade: funções podem ser passadas como argumentos, simplificadas, comparadas, etc;

O λ-cálculo é completo, mínimo e simples. Sua sintaxe é enxuta, contendo poucos tipos de expressões, assim como sua semântica, que também é de fácil compreensão.

Nas subseções seguintes, o λ-cálculo é descrito, tendo como referência [26].

2.1.1 Conceitos e exemplos

Nossa descrição do "λ-cálculo" começa com algumas motivações para a notação a ser utilizada. Uma função é um mapeamento de elementos de um domínio para elementos no contradomínio dado por uma regra. Por exemplo,

$$cube: \ Integer \rightarrow \ Integer \quad \text{onde} \quad cube(n) = n^3.$$

Podemos levantar certas questões em relação a definição de funções:

- Qual é o valor do identificador "$cube$"?
- Como podemos representar o objeto ligado a "$cube$"?
- Esta função pode ser definida sem dar-lhe um nome?

A notação lambda de Church permite a definição de função anônima, ou seja, uma função sem um nome:

$\lambda n.n^3$ define a função que mapeia cada n de um domínio para n^3.

Dizemos que a expressão representada por $\lambda n.n^3$ é o valor ligado ao identificador "$cube$". O número e a ordem dos parâmetros para uma função são especificados entre o símbolo λ e uma expressão. Por exemplo, a expressão $a^3 + b$ é ambígua como a definição de uma regra de função:

$$(3, 4) \vDash 3^3 + 4 = 31 \text{ ou } (3, 4) \vDash 4^3 + 3 = 67$$

A notação lambda resolve a ambigüidade especificando a ordem dos parâmetros:

$$\lambda a.\lambda b.a^3 + b \quad \text{ou} \quad \lambda b.\lambda a.a^3 + b.$$

2.1.2 Sintaxe

Como descrito anteriormente, o λ-cálculo é uma notação para definição de funções. As expressões desta notação são chamadas λ-expressões. Uma λ-expressão denota uma função. As λ-expressões são definidas a partir do alfabeto descrito abaixo.

Definição 2.1 (alfabeto). *Lambda expressões são palavras sobre o seguinte alfabeto:*

- ***Variáveis**: $x, y, z, x_1, y_1, z_1, ...$*
- ***Símbolos de Pontuação**: (.)*
- ***Conectivo**: λ*

As variáveis $x, y, z, x_1, y_1, z_1, ...$ formam uma coleção infinita. Notação: $V, V_1, V_2...$ são meta-variáveis que representam variáveis quaisquer.

As constantes $a, b, c, a_1, b_1, c_1...$ formam uma coleção infinita. Notação: $C, C_1, C_2...$ são meta-variáveis que representam constantes quaisquer.

O λ-cálculo puro não tem constantes e funções predefinidas, entretanto permite a definição de constantes comuns, funções da aritmética, manipulação de listas e outros. Em nossos exemplos, seguindo [26], constantes e funções predefinidas serão permitidas, incluindo numerais (por exemplo, 12), add (para adição), mul (para multiplicação), $succ$ (a função sucessor) e outras. Consideramos constantes e funções predefinidas que representam funções computáveis, as quais podem ser definidas no λ-cálculo puro.

Definição 2.2 (lambda expressões). *O conjunto das λ-expressões é definido por:*

1. *Se V é uma variável qualquer então V é uma λ-expressão;*
2. *Se E_1 e E_2 são λ-expressões então $(E_1\ E_2)$ é uma λ-expressão;*
3. *Se E é uma λ-expressão e V uma variável qualquer então $\lambda V.E$ é uma λ-expressão;*

Exemplos de λ-expressões:

- $(E\ \ x)$;

  - Pelo item 1 temos que x é uma λ-expressão. Sabendo que E é uma λ-expressão, pelo item 2 temos que $(E\ \ x)$ é uma λ-expressão.

- $\lambda x.((add\ \ 5)\ \ x)$;
  - Se add é uma λ-expressão que representa a função adição e considerando que 5 é uma λ-expressão que representa a constante 5, então pelo item 2 temos que $(add\ \ 5)$ é uma λ-expressão. Como x é uma λ-expressão temos que $((add\ \ 5)\ \ x)$ é uma λ-expressão. Finalmente, pelo item 3, temos que $\lambda x.((add\ \ 5)\ \ x)$ é uma λ-expressão.
- $\lambda x.(\lambda y.E)$;
  - Pelo item 3, temos que $\lambda y.E$ é uma λ-expressão. Logo, pelo item 3 $\lambda x.(\lambda y.E)$ é uma λ-expressão.
- $\lambda x.(\lambda y.f(x\ \ y))$;
  - Pelo item 1, temos que x e y são λ-expressões. Logo, pelo item 2 temos que $(x\ \ y)$ é uma λ-expressão. Se f é uma λ-expressão, então pelo item 2 $(f(x\ \ y))$ é uma λ-expressão. Pelo item 3 $\lambda y.f(x\ \ y)$ é uma λ-expressão. Logo, pelo item 3 $\lambda x.(\lambda y.f(x\ \ y))$ é uma λ-expressão.

Definição 2.3 (aplicação). *A λ-expressão $(E_1\ E_2)$ denota o resultado da aplicação da função E_1 em E_2.*

Exemplo: se (m, n) denota uma função representando o par de números m e n e sum denota a função adição, então a aplicação $(sum(m, n))$ denota o número $m + n$.

Definição 2.4 (abstração). *Se V é uma variável e E é uma λ-expressão, então $\lambda V.E$ é uma abstração com a variável ligada V e o escopo E.*

O significado da abstração pode ser compreendido da seguinte forma, se $f(x)$ representa uma expressão, possivelmente contendo a variável x, então representamos por $\lambda x.f(x)$ a função que associa a cada argumento a o valor $f(a)$.

Exemplo: Seja a função adiciona um dada por $sum(x, 1)$ então temos que $\lambda x.sum(x, 1)$ representa a função que recebe um argumento a e retorna como resultado a função $sum(a, 1)$.

Utilizando a Forma Normal de *Backus* (FNB) e considerando as constantes e funções predefinidas, a sintaxe das λ-expressões é dada por:

$$\begin{aligned} < \lambda\text{-expressao} > \quad ::= & < variavel > \\ & \mid \ < constante > \mid \ < funcoes > \\ & \mid \ (< \lambda\text{-expressao} >< \lambda\text{-expressao} >) \\ & \mid \ (\lambda < variavel > . < \lambda\text{-expressao} >) \end{aligned}$$

As λ-expressões serão ilustradas através de exemplos, utilizando a notação prefixa para operações binárias predefinidas.

- A λ-expressão $\lambda x.x$ denota a função identidade tal que: $((\lambda x.x)\ E) = E$ para qualquer λ-expressão E.

- A expressão $\lambda x.(add\ x\ 1)$ denota a função sucessor no conjunto dos inteiros.
  Logo, $(\lambda x.(add\ x\ 1))\ 5) = 6$. Note que a função "$add$" e as constantes 1 e 5 são λ-expressões predefinidas.

- A abstração $(\lambda f.(\lambda x.(f(f\ x))))$ descreve uma função com dois argumentos, o primeiro uma função e o segundo um valor, e aplica a função para o valor duas vezes. Se sqr é uma função (predefinida) que eleva seus argumentos ao quadrado, então

  $$\begin{aligned} (((\lambda f.(\lambda x.(f(f\ x))))\ sqr)\ 3) \quad & = ((\lambda x.(sqr\ (sqr\ x)))\ 3) \\ & = (sqr\ (sqr\ 3)) = (sqr\ 9) = 81 \end{aligned}$$

  Neste item f é trocado por sqr e depois x por 3.

Estes exemplos mostram que o número de parênteses nas λ-expressões pode aumentar muito. As seguintes convenções notacionais permitem abreviações que reduzem o número de parênteses:

1. Aplicação de função tem associação da esquerda para direita.

   $E_1\ \ E_2\ \ E_3$ equivale a $((E_1\ \ E_2)\ E_3)$

2. O escopo de "$\lambda < variavel >$" em uma abstração estende para direita o máximo possível.

$\lambda x.E_1\ \ E_2\ \ E_3$ equivale a $(\lambda x.(E_1\ \ E_2\ \ E_3))$ e não $((\lambda x.E_1\ \ E_2)\ E_3)$.

A aplicação tem maior precedência do que uma abstração, parênteses são necessários para $(\lambda x.E_1\ \ E_2)\ E_3$, pois neste caso, E_3 é um argumento para a função $\lambda x.E_1\ \ E_2$ e não é parte do escopo da função, como no exemplo acima.

3. Uma abstração permite que uma lista de variáveis abrevie uma série de abstrações lambda.

$\lambda x\ y\ z.E$ equivale a $(\lambda x.(\lambda y.(\lambda z.E)))$

Exemplo: $\lambda x\ y.add\ y\ x$ equivale a $(\lambda x.(\lambda y.((add\ y)\ x)))$.

2.1.3 Semântica das λ-expressões

Uma λ-expressão tem como significado a λ-expressão resultante após a avaliação de todas as aplicações de funções. A avaliação de uma lambda expressão é chamada de redução. A regra de redução básica envolve substituições das variáveis livres de forma similar a maneira como os parâmetros em uma função são passados como argumentos em uma chamada de função. Inicialmente, definimos os conceitos de ocorrências de variáveis livres e substituição de expressões por variáveis. Em seguida, definimos as regras de redução. Finalmente, descrevemos como representar objetos em λ-cálculo.

Definição 2.5 (variáveis livres e ligadas). *Uma ocorrência de uma variável x em uma λ-expressão é dita ligada se está dentro do escopo de "λx"; caso contrário é dita livre.*

Uma variável pode ocorrer tanto ligada como livre em um mesma lambda expressão; por exemplo, em $\lambda x.y\ \lambda y.y\ x$ a primeira ocorrência de y é livre e a segunda é ligada.

A notação $E\{x \leftarrow E_1\}$ refere-se a lambda expressão obtida pela substituição de cada ocorrência livre da variável x em E pela λ-expressão E_1. Tal substituição é chamada *segura* se nenhuma variável livre em E_1 torna-se ligada no resultado da substituição $E\{x \leftarrow E_1\}$.

Por exemplo, a substituição ingênua $(\lambda x.(mult\ y\ x))\{y \leftarrow x\}$ produz $(\lambda x.(mult\ x\ x))$. Esta substituição não é segura pois no lugar da variável y, que é livre aparece a variável x que é ligada. O resultado da substituição representa a operação de elevar ao quadrado, diferente da λ-expressão original. Não se pode permitir mudanças na semântica causadas por substituições como a ingênua. A semântica das λ-expressões deve ser preservada, após a aplicação de substituições seguras.

Definição 2.6 (conjunto das variáveis livres). *O conjunto das variáveis que ocorrem livres em uma expressão E, denotado por $FV(E)$, é definido por:*

a) $FV(C) = \emptyset$ *para qualquer constante* C*;*

b) $FV(x) = \{x\}$ *para qualquer variável* x*;*

c) $FV(E_1\ \ E_2) = FV(E_1) \cup FV(E_2)$

d) $FV(\lambda x.E) = FV(E) - \{x\}$

Uma λ-expressão E com nenhuma variável livre $(FV(E) = \emptyset)$ é denominada *fechada*.

Definição 2.7 (substituição). *A substituição de uma variável livre por uma expressão em uma λ-expressão é denotada por $E\{x \leftarrow E_1\}$ e é definida da seguinte maneira:*

- $V\{V \leftarrow E_1\} = E_1$
- $V_1\{V_2 \leftarrow E_1\} = V_1$ *onde* $V_1 \neq V_2$
- $C\{V \leftarrow E_1\} = C$ *onde* C *é uma constante qualquer*
- $(E_1\ \ E_2)\{V \leftarrow E\} = ((E_1\{V \leftarrow E\})(E_2\{V \leftarrow E\}))$
- $(\lambda V.E)\{V \leftarrow E_1\} = (\lambda V.E)$
- $(\lambda V_1.E)\{V_2 \leftarrow E_1\} = \lambda V_1.(E\{V_2 \leftarrow E_1\})$ *quando* $V_1 \neq V_2$ *e* $V_1 \notin FV(E_1)$
- $(\lambda V_1.E)\{V_2 \leftarrow E_1\} = \lambda V_3.E\{V_1 \leftarrow V_3\}\{V_2 \leftarrow E_1\}$ *quando* $V_1 \neq V_2$ *e* $V_1 \in FV(E_1)$*, onde* $V_3 \neq V_2$ *e* $V_3 \notin FV(E\ \ E_1)$.

Em seguida, as regras para simplificação de λ-expressões são definidas com base nesta idéia de substituição.

Definição 2.8 (α-redução ou α-conversão). *Qualquer abstração da forma $\lambda V.E$ pode ser convertida para $\lambda V_1.E\{V \leftarrow V_1\}$ desde que a substituição de V por V_1 em E seja segura.*
Notação: $\lambda V.E \xrightarrow{\alpha} \lambda V_1.E\{V \leftarrow V_1\}$

Exemplos:

$$\lambda x.x \xrightarrow{\alpha} \lambda y.y$$
$$\lambda x.f\ x \xrightarrow{\alpha} \lambda y.f\ y$$

Temos que

$$\lambda x.\lambda y.add\ x\ y \longrightarrow \lambda y.\lambda y.add\ y\ y$$

não é uma α-conversão, pois a substituição $(\lambda y.add\ x\ y)\{x \leftarrow y\}$ não é segura, visto que a variável y substituída no lugar de x torna-se ligada no resultado.

Definição 2.9 (β-redução ou β-conversão). *Qualquer aplicação da forma $(\lambda V.E_1)\ E_2$ pode ser convertida para $E_1\{V \leftarrow E_2\}$ desde que a substituição de V por E_2 em E_1 seja segura.*
Notação: $(\lambda V.E_1)\ E_2 \xrightarrow{\beta} E_1\{V \leftarrow E_2\}$.

Exemplos:

$$(\lambda x.f\ x)\ E \xrightarrow{\beta} f\ E$$
$$(\lambda x.(\lambda y.add\ x\ y))\ 3 \xrightarrow{\beta} \lambda y.add\ 3\ y$$
$$(\lambda y.add\ 3\ y)\ 4 \xrightarrow{\beta} add\ 3\ 4$$

Temos que

$$(\lambda x.(\lambda y.add\ x\ y))\ (square\ y) \longrightarrow \lambda y.add\ (square\ y)\ y$$

não é uma β-conversão, pois a substituição $(\lambda y.add\ x\ y)\{x \leftarrow (square\ y)\}$ não é válida, visto que a variável y está livre em $(square\ y)$ mas torna-se ligada depois da substituição.

O lado esquerdo $(\lambda V.E_1)\ E_2$ de uma β-redução é denominado $\beta-$*redex*: termo derivado das palavras "redução de expressão" e significa que uma expressão pode ser β-reduzida. A β-redução serve como a principal regra de avaliação no λ-cálculo. A α-redução é simplesmente utilizada para fazer substituições (renomeação) de variáveis válidas.

A avaliação de uma λ-expressão consiste de uma série de β-reduções possivelmente intercaladas com α-reduções (renomeação de variáveis ligadas). Assuma que $E \rightarrow E_1$ equivale a $E \xrightarrow{\beta} E_1$ ou $E \xrightarrow{\alpha} E_1$ e seja $\xrightarrow{*}$ um fecho reflexivo e transitivo de $\rightarrow$. Disto, temos que para quaisquer λ-expressões E, E_1, E_2 e E_3, $E \xrightarrow{*} E$ e $(E_1 \xrightarrow{*} E_2$ e $E_2 \xrightarrow{*} E_3)$ implica em $E_1 \xrightarrow{*} E_3$. O objetivo de uma avaliação em λ-cálculo é reduzir uma λ-expressão via $\rightarrow$ até que esta não contenha mais β-redexes.

Definição 2.10 (η-redução ou η-conversão). *Qualquer abstração da forma $\lambda V.(E\ V)$ na qual V não tem ocorrência livre em E pode ser reduzida para E.*
Notação: $\lambda V.(E\ V) \xrightarrow{\eta} E$.

Exemplos:

$$\lambda x.add\ x \xrightarrow{\eta} add$$
$$\lambda y.add\ x\ y \xrightarrow{\eta} add\ x$$

Temos que

$$\lambda x.add\ x\ x \longrightarrow add\ x$$

não é uma η-conversão segura, pois a variável x está livre em $add\ x$.

Em λ-cálculo aplicado, contendo operações e valores predefinidos, precisamos de uma justificação para avaliação das combinações de constantes. Isto é feito através de uma regra de redução denominada δ-redução.

Definição 2.11 (δ-redução). *Se o λ-cálculo tem constantes e funções predefinidas (isto é, não é puro), regras associadas aos valores e funções predefinidos são chamadas δ-regras.*
Notação: $E_1 \xrightarrow{\delta} E_2$ significa que E_2 é obtido pela aplicação de uma δ-regra em alguma subexpressão de E_1.

Exemplos:

$$(add\ \ 3\ \ 5) \xrightarrow{\delta} 8$$
$$(not\ \ true) \xrightarrow{\delta} false$$

Observação. Quando adicionamos constantes e regras no λ-cálculo devemos ter o cuidado de não destruir as propriedades do λ-cálculo, ou seja, invalidar o teorema de Church-Rosser (descrito adiante). Em [4] encontramos as condições que devem ser satisfeitas (ou seja, aquelas que preservam o teorema de Church Rosser) quando uma δ-regra é adicionada.

2.1.4 Representando objetos em λ-cálculo

As funções recursivas formam uma classe importante das funções numéricas. Logo após Church ter inventado o λ-cálculo, Kleene provou que toda função recursiva pode ser representada no λ-cálculo [32]. Uma descrição mais detalhada da representação de funções recursivas em λ-cálculo pode ser encontrada em [26].

Nesta seção, a representação em λ-cálculo de objetos de dados (números), estruturas de dados (pares) e algumas funções úteis é descrita. A idéia é codificar estes objetos e estruturas de forma que eles não violem suas propriedades. Por exemplo, para representar os valores verdades *true* e *false* e a função booleana *not*, as λ-expressões são descritas de forma que as propriedades abaixo sejam satisfeitas:

$$not\ true\ =\ false$$
$$not\ false\ =\ true$$

Para definição de uma λ-expressão, utilizaremos a seguinte notação:

$$define < nome > = < \lambda\text{-expressao} >$$

Valores verdades e a condicional

Esta seção define as λ-expressões $true$, $false$, not e $(E \rightarrow E_1|E_2)$ com as seguintes propriedades:

$$not\ true\ =\ false$$
$$not\ false\ =\ true$$

$$(true \rightarrow E_1|E_2) = E_1$$
$$(false \rightarrow E_1|E_2) = E_2$$

As λ-expressões $true$ e $false$ representam os valores verdades true e false e not representa a função negação. A λ-expressão $(E \rightarrow E_1|E_2)$ representa a condicional "se E então E_1 senão E_2".

$$\begin{aligned} define\ true &= \lambda x.\lambda y.x \\ define\ false &= \lambda x.\lambda y.y \\ define\ not &= \lambda x.x\ false\ true \end{aligned}$$

Utilizamos as regras de redução para mostrar que estas definições atendem as propriedades desejadas. Por exemplo:

$$\begin{aligned} not\ true &= (\lambda x.x\ false\ true)\ true\ (\text{definição de not}) \\ &= (true\ false\ true)\ (\beta\text{-redução}) \\ &= (\lambda x.\lambda y.x)\ false\ true\ (\text{definição de } true) \\ &= (\lambda y.false)\ true\ (\beta - \text{redução}) \\ &= false\ (\beta - \text{redução}) \end{aligned}$$

Expressões condicionais $(E \rightarrow E_1|E_2)$ podem ser definidas da seguinte forma:

$$define\ (E \rightarrow E_1|E_2) = (E\ E_1\ E_2)$$

A notação condicional comporta-se como deve:

$$\begin{aligned} (true \rightarrow E_1|E_2) &= true\ E_1\ E_2 \\ &= (\lambda x.\lambda y.x)\ E_1\ E_2 \\ &= E_1 \end{aligned}$$

e

$$\begin{aligned} (false \rightarrow E_1|E_2) &= false\ E_1\ E_2 \\ &= (\lambda x.\lambda y.y)\ E_1\ E_2 \\ &= E_2 \end{aligned}$$

Neste trabalho, não demonstraremos a validade das demais λ-expressões usadas, como foi feito para not, $true$ e $(E \rightarrow E_1|E_2)$. Em [26] encontram-se tais demonstrações.

Pares e tuplas

As seguintes definições representam pares e n-tuplas em λ-cálculo.

$$\begin{aligned} define\ \ fst &= \lambda x.x\ \ true \\ define\ \ snd &= \lambda x.x\ \ false \\ define\ \ (E_1, E_2) &= \lambda f.f\ \ E_1\ \ E_2 \end{aligned}$$

(E_1, E_2) é uma λ-expressão representando um par ordenado cujo primeiro componente (E_1) é acessado com a função fst (primeiro) e o segundo componente (E_2) é acessado com a função snd (segundo).

Um par é uma estrutura de dados com dois componentes. A generalização para n componentes é chamada n-tupla e é definida em termos de pares.

$$define\ \ (E_1, E_2, \ldots, E_n)\ =\ (E_1, (E_2, (\ldots (E_{n-1}, E_n) \ldots)))$$

$(E_1, \ldots, E_n)$ é uma n-tupla com os componentes $E_1, \ldots, E_n$ e tamanho n.

Números e operações aritméticas básicas

Existem várias maneiras para representar números através de λ-expressões, cada qual com suas vantagens e desvantagens. Neste trabalho utilizaremos os numerais de Church [26]. O objetivo é definir para cada número n uma λ-expressão $\underline{n}$ que o represente, bem como definir λ-expressões que representem as operações primitivas da aritmética. Por exemplo, as λ-expressões suc,add e $iszero$, definidas em seguida, representam a função sucessor $(n \mapsto n+1)$, a função adição e o teste "é zero", respectivamente.

$$\begin{aligned} define\ \ \underline{0} &= \lambda f\ \ x.x \\ define\ \ \underline{1} &= \lambda f\ \ x.f\ \ x \\ define\ \ \underline{2} &= \lambda f\ \ x.f(f\ \ x) \\ &\vdots \\ define\ \ \underline{n} &= \lambda f\ \ x.f^n\ x \\ &\vdots \end{aligned}$$

Abaixo, a definição das operações aritméticas.

$$
\begin{aligned}
define\ suc &= \lambda y\ f\ x.y\ f(f\ x) \\
define\ add &= \lambda z\ y\ f\ x.z\ f(y\ f\ x) \\
define\ iszero &= \lambda y.y(\lambda x.false)\ true
\end{aligned}
$$

Listas

Uma lista é definida em λ-cálculo recursivamente e é representada por um par (b, E) onde b representa um booleano que indica se a lista é vazia ou não. Então, um teste $null$ pode ser representado por inspecionar o primeiro componente:

$$define\ null\ =\ fst$$

Para evitar confusão entre λ-expressões representando pares e aquelas que representam as listas, a notação $[E_1; \ldots; E_n]$ é utilizada para representar uma lista cujos componentes são $E_1; \ldots; E_n$.

Uma lista é definida da seguinte forma:

$$
\begin{aligned}
define\ [\ ] &= (true, \bot) \\
define\ [E] &= (false, (E), [\ ]) \\
define\ [E_1; E_2] &= (false, (E_1, [E_2])) \\
&\vdots \\
define\ [E_1; \ldots; E_n] &= (false, (E_1, [E_2; \ldots; E_n]))
\end{aligned}
$$

Assume-se que a λ-expressão $\bot$ representa uma "função indefinida". Esta pode ser definida em λ-cálculo utilizando um operador de ponto fixo [26].

Algumas operações relacionadas as listas como o primeiro elemento (hd), a cauda (tl) e a construção ($cons$) destas são definidas em seguida:

$$
\begin{aligned}
define\ hd &= \lambda x.(null\ x \rightarrow \bot \mid fst(snd\ x)) \\
define\ tl &= \lambda x.(null\ x \rightarrow \bot \mid snd(snd\ x)) \\
define\ cons &= \lambda x\ y.(false, (x, y))
\end{aligned}
$$

2.1.5 Propriedades teóricas

O principal objetivo em manipular uma λ-expressão é reduzir esta para uma "forma mais simples".

Definição 2.12 (forma normal). *Uma λ-expressão está na forma normal se esta não contém nenhuma β-redexes (e nenhuma δ-regra em λ-cálculo aplicado).*
Isto significa, que uma λ-expressão na forma normal, não pode ser reduzida mais utilizando a β-redução ou a δ-regra. Uma expressão na forma normal não tem mais aplicações de funções a serem executadas.

Os conceitos de forma normal e estratégia de redução podem ser investigados ao responder as quatro questões seguintes:

- Toda λ-expressão pode ser reduzida para uma forma normal?
- Existe mais que um caminho para reduzir uma λ-expressão particular?
- Se existe mais que uma estratégia de redução, cada uma destas leva a mesma expressão na forma normal?
- Existe uma estratégia de redução que garantirá que uma expressão na forma normal será obtida?

A primeira questão tem uma resposta negativa, como mostrado pela λ-expressão $(\lambda x.x\ x)\ (\lambda x.x\ x)$. Esta β-redex reduz em si mesma, mostrando que o único caminho de redução nunca termina.

$$\begin{aligned}(\lambda\mathbf{x}.x\ x)\ \mathbf{(\lambda x.\ x\ x)} &\xrightarrow{\beta} (\lambda\mathbf{x}.x\ x)\ \mathbf{(\lambda x.\ x\ x)}\\ &\xrightarrow{\beta} (\lambda\mathbf{x}.x\ x)\ \mathbf{(\lambda x.\ x\ x)}\\ &\xrightarrow{\beta} \ \ldots\end{aligned}$$

A segunda pergunta tem uma resposta afirmativa, como mostra os exemplos abaixo.

Exemplo: $(\lambda x.\lambda y.(add\ \ y((\lambda z.(mult\ \ x\ \ z))3)))\ \ 7\ \ 5$
Caminho 1: $(\lambda\mathbf{x}.\lambda y.(add\ \ y((\lambda z.(mult\ \ x\ \ z))3)))\ \ \mathbf{7}\ \ 5$

$$\begin{aligned}&\xrightarrow{\beta} (\lambda\mathbf{y}.(add\ \ y((\lambda z.(mult\ \ 7\ \ z))3)))\ \ \mathbf{5}\\ &\xrightarrow{\beta} (add\ \ 5((\lambda\mathbf{z}.(mult\ \ 7\ \ z))\mathbf{3}))\\ &\xrightarrow{\beta} (add\ \ 5(\mathbf{mult\ \ \ 7\ \ \ 3})) \xrightarrow{\beta} (\mathbf{add\ \ \ 5\ \ \ 21)} \xrightarrow{\beta} 26\end{aligned}$$

Caminho 2: $(\lambda x.\lambda y.(add\ \ y((\lambda\mathbf{z}.(mult\ \ x\ \ z))\mathbf{3})))\ \ 7\ \ 5$

$$\xrightarrow{\beta} (\lambda\mathbf{x}.\lambda y.(add\ \ y(mult\ \ x\ \ 3)))\ \ \mathbf{7}\ \ 5$$
$$\xrightarrow{\beta} (\lambda\mathbf{y}.(add\ \ y(\mathbf{mult}\ \ \mathbf{7}\ \ \mathbf{3})))\ \ 5$$
$$\xrightarrow{\beta} (\lambda\mathbf{y}.(add\ \ y\ \ 21))\ \ \mathbf{5} \xrightarrow{\beta} (\mathbf{add}\ \ \mathbf{5}\ \ \mathbf{21}) \xrightarrow{\beta} 26$$

Neste exemplo ambos os caminhos levam ao mesmo resultado, o qual está na forma normal. Entretanto, existe λ-expressão que pode ser reduzida por caminhos diferentes os quais levam a resultados diferentes, veja o exemplo abaixo.

Exemplo: $(\lambda y.5)\ \ ((\lambda x.x\ \ x)\ \ (\lambda x.x\ \ x))$
Caminho 1: $(\lambda\mathbf{y}.5)$ **((**λ**x.x x) (**λ**x.x x))** $\xrightarrow{\beta} 5$

Caminho 2: $(\lambda y.5)\ \ ((\lambda\mathbf{x}.x\ \ x)\ \ \mathbf{(}\lambda\mathbf{x.x}\ \ \mathbf{x)})$

$$\xrightarrow{\beta} (\lambda y.5)\ \ ((\lambda\mathbf{x}.x\ \ x)\ \ \mathbf{(}\lambda\mathbf{x.x}\ \ \mathbf{x)})$$
$$\xrightarrow{\beta} (\lambda y.5)\ \ ((\lambda\mathbf{x}.x\ \ x)\ \ \mathbf{(}\lambda\mathbf{x.x}\ \ \mathbf{x)})\ \ \ldots$$

Neste exemplo o primeiro caminho reduz a "redex mais a esquerda" primeiro (ordem normal), levando para uma expressão na forma normal. Enquanto o segundo caminho avalia a aplicação "mais a direita" a cada passo (ordem aplicativa), resultando numa execução interminável.

As duas questões restantes podem ser respondidas aplicando os resultados provados por Alonzo Church e J. Barkley em 1936 [15].

Teorema 2.1 (Teorema de Church-Rosser I). *Para quaisquer λ-expressões E,E_1 e E_2, se $E \xrightarrow{*} E_1$ e $E \xrightarrow{*} E_2$, então existe uma λ-expressão E_3 tal que $E_1 \xrightarrow{*} E_3$ e $E_2 \xrightarrow{*} E_3$.*

Demonstração: A demonstração formal deste teorema pode ser encontrada em [4] ou [36].

Corolário 2.1. *Sejam E, E_1 e E_2 λ-expressões quaisquer, se $E \xrightarrow{*} E_1$ e $E \xrightarrow{*} E_2$, onde E_1 e E_2 estão na forma normal, então E_1 varia em relação a E_2 apenas na α-redução.*

Demonstração: A única redução possível para uma expressão na forma normal é uma α-redução. Portanto a λ-expressão E_3 no teorema deve ser uma variação de E_1 e E_2 somente por uma α-redução.

Este corolário diz que se uma λ-expressão tem uma forma normal, essa forma normal é única diferindo apenas nas variáveis ligadas, respondendo portanto a terceira questão.

Teorema 2.2 (Teorema de Church-Rosser II). *Sejam E, e E_1 λ-expressões quaisquer, se $E \xrightarrow{*} E_1$ onde E_1 está na forma normal, então existe uma redução de ordem normal de E para E_1.*

Demonstração: Veja novamente em [4].

O segundo teorema de Church-Rosser responde a quarta questão por dizer que a redução de ordem normal produzirá uma λ-expressão na forma normal se esta existe.

Uma redução de ordem normal pode ter uma das seguintes consequências:

1. Esta alcança uma única λ-expressão na forma normal.

2. Esta nunca termina.

Infelizmente, não existe algoritmo que determine para uma λ-expressão qualquer quais dessas consequências irá ocorrer.

Máquinas de Turing são máquinas abstratas projetadas na década de 30 por Alan Turing para modelar funções computáveis. Neste sentido, tem sido mostrado que o λ-cálculo é equivalente as Máquinas de Turing de forma que toda λ-expressão tem uma função equivalente definida por alguma Máquina de Turing e vice-versa. Esta equivalência dá credibilidade para a tese de Church.

Tese de Church. *As funções efetivamente computáveis nos inteiros positivos são precisamente as funções definíveis em λ-cálculo puro (e computáveis pelas Máquinas de Turing)* [26].

Alan Turing provou um resultado fundamental, denominado de indecidibilidade do problema da parada, o qual afirma que não existe algoritmo para determinar se uma Máquina de Turing irá parar ou não. Portanto, existem λ-expressões pelas quais não é possível determinar se uma redução de ordem normal irá terminar [15, 26].

Capítulo 3

Estrutura Funcional para RI baseada no λ-cálculo

Em [39] foi apresentado um meta-modelo para recuperação de informação, chamado Estrutura Funcional para RI. Esta estrutura permite representar, construir, combinar e comparar modelos de RI de forma algébrica sem realizar experimentos. Nesta estrutura os modelos de RI são representados através de funções. Como descrito no capítulo 2, o λ-cálculo apresenta uma notação expressiva e precisa para definição de funções. Propomos neste trabalho uma Estrutura Funcional para RI baseada no λ-cálculo. Formalmente, definimos os componentes tais como documentos, consultas, função de *ranking* ou função de similaridade de modelos de RI utilizando a notação do λ-cálculo.

Neste *framework* um dos principais objetivos é ter como elemento central da representação o conceito de função. Esta estratégia difere dos modelos tradicionais, que geralmente consideram pesos de termos, vetores, etc como fundamentos.

Como mostraremos neste trabalho, vários modelos clássicos de RI podem ser representados utilizando a notação funcional. Além disso, na notação do λ-cálculo as funções são representadas em um formato equacional. Assim, ao especificarmos as funções no λ-cálculo, a identificação dos argumentos e valores das funções ficam mais claros. Com isso, facilitamos o entendimento na representação e caracterização dos modelos de RI.

O *framework* proposto permite mostrar a noção de equivalência entre modelos. A passagem dos modelos de RI para o *framework* funcional facilita a comparação e combinação entre eles, pois os

modelos são representados utilizando a mesma notação funcional, o λ-cálculo. Além disso, esta representação permite a identificação e caracterização das funções presentes nos modelos de RI. Como descrito no Capítulo 2, toda função computável pode ser representada em λ-cálculo. Portanto, o *framework* proposto pode representar os modelos de RI que são expressos por funções computáveis. Sendo assim, o meta-modelo é caracterizado por sua generalidade e pelo formalismo descrito através da notação do λ-cálculo para definição das funções.

3.1 Fundamentos da Estrutura Funcional

Um modelo de recuperação de informação representa documentos e consultas para predizer o que um usuário considera relevante para sua necessidade de informação. São três os modelos clássicos seguidos por sistemas de RI para determinar a relevância de documentos: booleano, vetorial e probabilístico [3].

Os modelos clássicos, utilizados no processo de recuperação de informação, apresentam estratégias de busca de documentos similares à consulta. Estes modelos consideram que cada documento é descrito por um conjunto de termos, considerados como mutuamente independentes. Associa-se a cada par constituído por seu termo k_i e um documento d_j, um peso $w_{i,j} \geq 0$. Este peso quantifica a importância do termo k_i no documento d_j. Analogamente a cada par termo-consulta (k_i, q) associa-se o peso $w_{i,q}$. Finalmente, os modelos de RI definem uma função de similaridade ou função de *ranking* que denota o grau de relevância dos documentos para uma consulta. No capítulo 4, descrevemos estes modelos.

Apresentamos nesta seção as definições da estrutura funcional utilizando a notação do λ-cálculo. Os fundamentos são divididos em representação e comparação de modelos.

3.1.1 Representação de Modelos

Para representar modelos de RI na estrutura funcional, definimos os componentes: termo funcional, função peso, universo funcional, função similaridade entre dois objetos funcionais, função de documentos relevantes, coleção de referência funcional, casamento entre documentos e consultas funcionais.

Estes componentes são representados através de funções. Para definirmos estas funções utilizamos a notação λ-cálculo. O capítulo 2, descreve várias funções na notação λ-cálculo tais como listas, os valores verdades, numerais, operações básicas e pares. Neste trabalho, as outras funções definidas serão consideradas como constantes ou funções predefinidas no alfabeto do λ-cálculo. Em particular, os termos de uma coleção de documentos serão considerados constantes predefinidas no λ-cálculo. Além disso, alguns objetos como conjuntos são representados através de listas.

Definição 3.1 (Termo Funcional). *Um termo funcional tf é uma λ-expressão cuja semântica relaciona um conjunto de termos. O termo funcional tf é dado por:*

$$\lambda x.E.$$

Exemplo: Seja **K** = $\{k_1, ..., k_t\}$ um conjunto de termos e $L_{\mathbf{K}}$ o conjunto de todas as listas formadas com elementos de **K**. A função $syn : \mathbf{K} \rightarrow L_{\mathbf{K}}$ é a função sinonímia tal que, dado um termo, retorna a lista de sinônimos de cada termo.

$$\text{syn: } \lambda x.syn\ x.$$

Logo temos que: $(\lambda x.syn\ x)\ (k_i) = (syn\ x)\{x \leftarrow k_i\} = [k_{i1}; ...; k_{is}]$.

Dado **K** um conjunto de termos, qualquer λ-expressão cujo domínio é **K** é um termo funcional. Então, um termo funcional é uma λ-expressão que expressa qualquer relação entre os termos, sendo esta uma importante ferramenta para modelagem de problemas em RI. Neste contexto, o foco principal são as funções que relacionam os termos do conjunto **K** e não os termos propriamente ditos.

Exemplo: A λ-expressão $peso$ é um exemplo de termo funcional. Uma λ-expressão $peso$ é uma função cujo resultado é o peso do termo em um documento ou em uma consulta. Seja $\mathbf{C} = \{d_1, \ldots, d_n\}$ uma coleção de documentos, $\mathbf{K} = \{k_1, \ldots, k_t\}$ um conjunto de termos de **C**, e q uma consulta. A função peso $peso : \mathbf{K} \times \{\mathbf{C} \cup \{q\}\} \rightarrow \mathbb{R}$ é tal que $peso(k_i, d_j)$ retorna o peso do termo k_i no documento d_j e $peso(k_i, q)$ retorna o peso do termo k_i na consulta q. A λ-expressão peso é dada por:

$$\text{peso: } \lambda x \lambda y.peso(x, y).$$

Logo, temos que: $(\lambda x \lambda y.peso(x,y))\ \ (k_i, d_j) = peso(x,y)\{x \leftarrow k_i\}\{y \leftarrow d_j\} = peso(k_i, d_j)$.

Para simplificar, usamos a seguinte notação: seja $peso_j : \mathbf{K} \rightarrow \mathbb{R}$ uma função unária que retorna o peso de um termo no documento d_j. A λ-expressão $peso_j$ é dada por: $\lambda x.peso(d_j, x)$. Quando aplicada ao termo k_i retorna o peso do termo k_i no documento d_j. Analogamente, seja $peso_q : \mathbf{K} \rightarrow \mathbb{R}$ uma função unária que retorna o peso de um termo na consulta q. A λ-expressão $peso_q$ é dada por: $\lambda x.peso(q, x)$. Quando aplicada ao termo k_i retorna o peso do termo k_i na consulta q. As λ-expressões $peso_j$ e $peso_q$ são termos funcionais.

Definição 3.2 (Documento funcional). *Um documento funcional df_j é uma λ-expressão da seguinte forma:*

$$[tf_1; ...; tf_n]$$

ou seja, df_j é representado por uma lista de termos funcionais $[tf_1; ...; tf_n]$ que relacionam os termos do documento d_j e tf_i denota o i-ésimo termo funcional.

Exemplo: No modelo vetorial temos que $df_j = [peso_j]$.

Definição 3.3 (Consulta funcional). *Uma consulta funcional qf é uma λ-expressão da seguinte forma:*

$$[tf_1; ...; tf_n]$$

ou seja, qf é representada por uma lista de termos funcionais $[tf_1; ...; tf_n]$ que relacionam os termos da consulta q e tf_i denota o i-ésimo termo funcional.

Exemplo: No modelo vetorial temos que $qf = [peso_q]$.

Definição 3.4 (Universo funcional). *A lista dos documentos funcionais e consultas funcionais formam o universo funcional U_f, que é denotado por $[df_1, ..., df_n, qf_1, ..., qf_m]$. Os objetos funcionais deste universo são representados por uma lista de termos funcionais.*

Definição 3.5 (Função similaridade). *Dado o universo funcional U_f, a similaridade Δ é uma λ-expressão da seguinte forma:*

$$\lambda x \lambda y.sim(x, y)$$

tal que $\Delta : U_f \times U_f \rightarrow \Re$. *Portanto, para* $(of_i, of_j) \in U_f \times U_f$ *temos que* $\Delta(of_j, of_i) \in \Re$.

Em geral, a função similaridade apresenta as seguintes propriedades:

1. $0 \leq \Delta(of_j, of_i) \leq 1$ (normalização)
2. $\Delta(of_j, of_j) = 1$ (reflexividade)
3. $\Delta(of_j, of_i) = \Delta(of_i, of_j)$(simetria)

Notação:

- Se Δ satisfaz a normalização ela é do tipo n, denotada por Δ_n
- Se Δ satisfaz a reflexividade ela é do tipo r, denotada por Δ_r
- Se Δ satisfaz a simetria ela é do tipo s, denotada por Δ_s
- Se Δ satisfaz a normalização, a reflexividade e a simetria ela é do tipo nrs, denotada por Δ_{nrs}

A propriedade de normalização da função similaridade é importante para a combinação e comparação entre modelos, e as propriedades de reflexividade e simetria são importantes por exemplo, para clusterização de documentos. Além disso, se a similaridade não é reflexiva é difícil montar um *ranking*.

Definição 3.6 (Modelo Funcional). *Um modelo funcional é definido pela tupla*

$$\Psi = \langle \boldsymbol{D}_f, \boldsymbol{Q}_f, \Delta \rangle$$

onde:

- $\boldsymbol{D}_f$ *é a lista de documentos funcionais* $[df_1; \ldots; df_n]$
- $\boldsymbol{Q}_f$ *é a lista de consultas funcionais* $[qf_1, \ldots, qf_m]$

- Δ *é uma função similaridade.*

onde n e m são o número de documentos funcionais e o número de consultas funcionais da coleção de referência, respectivamente.

Definição 3.7 (Função de documentos relevantes). *A função de documentos relevantes ou conjunto ideal é uma função que dado uma consulta funcional e o conjunto de documentos funcionais (*$\boldsymbol{D}_f$*) retorna o conjunto de documentos relevantes. Seja* $L_{\boldsymbol{D}_f}$ *o conjunto de todas as listas formadas com elementos de* $\boldsymbol{D}_f$*. A função de documentos relevantes é definida por* $I : \boldsymbol{Q}_f \times \boldsymbol{D}_f \rightarrow L_{\boldsymbol{D}_f}$*.*

Definição 3.8 (Coleção de Referência Funcional). *A coleção de referência funcional,* $\boldsymbol{C}_f$*, é formada pelo universo funcional* $\boldsymbol{U}_f$ *(a lista dos documentos funcionais e consultas funcionais* $[df_1, \ldots, df_n, qf_1, qf_2, \ldots, qf_m]$*) e pela função de documentos relevantes* $I : \boldsymbol{Q}_f \times \boldsymbol{D}_f \rightarrow L_{\boldsymbol{D}_f}$ *ou conjunto ideal para as consultas funcionais. Portanto,* $\boldsymbol{C}_f = \langle \boldsymbol{U}_f, I \rangle$

Definição 3.9 (Função de Recuperação). *Seja* $L_{\boldsymbol{D}_f}$ *o conjunto de todas as listas formadas com elementos de* $\boldsymbol{D}_f$*. A função de recuperação retorna a lista de documentos ordenados (ou ranking) de acordo com a função de similaridade (*Δ*) que são considerados relevantes para a consulta. Esta função é definida por* $Rank : \boldsymbol{Q}_f \times \boldsymbol{D}_f \rightarrow L_{\boldsymbol{D}_f}$*.*

Definição 3.10 (Casamento entre Documentos e Consultas Funcionais). *A função similaridade* Δ *define um ranking cuja ordenação é decrescente. Seja* df_j *um documento funcional e* qf *uma consulta funcional. Seja* α *um número positivo, tal que* $0 \leq \alpha \leq 1$*. Dado um limite inferior* α*, o casamento entre* df_j *e* qf *conforme* Δ *ocorre se* $\Delta(qf, df_j) \geq \alpha$*, onde* Δ *é uma função similaridade.*

3.1.2 Comparação de Modelos

Com a representação de modelos de RI na estrutura funcional, podemos verificar a equivalência entre eles. Assim, dados dois modelos Ψ^a e Ψ^b é possível dizer se são equivalentes ou não. Define-se uma relação de comparação entre modelos: a relação de equivalência indicada a seguir [39].

Definição 3.11 (Equivalência entre Modelos Funcionais em Relação a uma Consulta). *Dois modelos funcionais* $\Psi^a = \langle [df_1^a, ..., df_n^a], [qf^a], \Delta^a \rangle$ *e* $\Psi^b = \langle [df_1^b, ..., df_n^b], [qf^b], \Delta^b \rangle$ *são equivalentes em relação à uma consulta* qf*, se e somente se existe uma função bijetora* $\phi : \{df_1^a, ..., df_n^a\} \rightarrow$

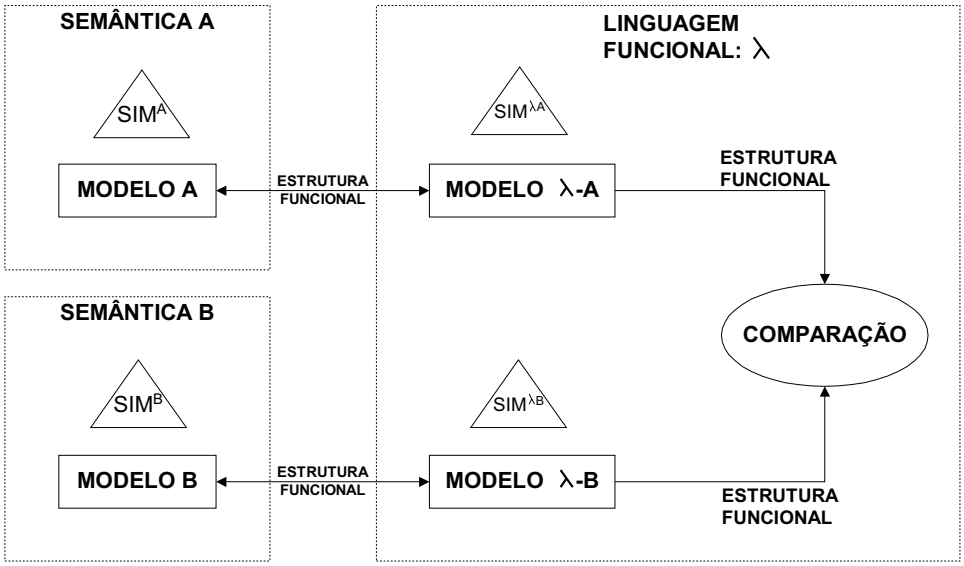


Fig. 3.1: Esquema geral para comparação de equivalência entre modelos de RI

$\{df_1^b, ..., df_n^b\}$, *tal que se* $\phi(df_i^a) = df_i^b$ *e* $\phi(df_k^a) = df_k^b$, *então as duas condições abaixo são satisfeitas:*

1. $\Delta^a(qf^a, df_i^a) = \Delta^a(qf^a, df_k^a) \Leftrightarrow \Delta^b(qf^b, df_i^b) = \Delta^b(qf^b, df_k^b)$
2. $\Delta^a(qf^a, df_i^a) > \Delta^a(qf^a, df_k^a) \Leftrightarrow \Delta^b(qf^b, df_i^b) > \Delta^b(qf^b, df_k^b)$

Considerando que o casamento entre um documento funcional df_j e uma consulta funcional qf ocorre se $\Delta(df_j, qf)$ for maior que um dado limite inferior, então as duas propriedades devem ser satisfeitas. A propriedade 1 garante que se dois documentos funcionais (df_i^a e df_k^a) possuem a mesma similaridade em relação à uma consulta funcional para o modelo Ψ^a, então os mesmos documentos representados no modelo Ψ^b (df_i^b e df_k^b) também possuem similaridade iguais em relação à uma consulta funcional para o modelo Ψ^b, e vice-versa. A propriedade 2 garante que se um documento funcional (df_i^a) possui similaridade maior que outro documento (df_k^a) em relação a consulta qf para o modelo Ψ^a, então o primeiro documento (df_i^a) representado no modelo Ψ^b, por df_i^b, possui similaridade maior que o segundo documento (df_k^a) representado no modelo Ψ^b, por df_k^b. Isto garante que a ordenação do $ranking$ seja a mesma. Com estas duas propriedades satisfeitas, garante-se que os modelos Ψ^a e Ψ^b geram o mesmo $ranking$.

Definição 3.12 (Equivalência entre Modelos Funcionais). *Dois modelos funcionais* $\Psi^a = \langle [df_1^a, ..., df_n^a], \boldsymbol{Q}_f^a, \Delta^a \rangle$ *e* $\Psi^b = \langle [df_1^b, ..., df_n^b], \boldsymbol{Q}_f^b, \Delta^b \rangle$ *são equivalentes em relação a um conjunto de consultas* $\boldsymbol{Q}_f = \boldsymbol{Q}_f^a \cup \boldsymbol{Q}_f^b$ *se e somente se para toda consulta funcional* $qf \in \boldsymbol{Q}_f$, Ψ^a *equivale a* Ψ^b *em relação à consulta* qf.

Essas condições garantem que os modelos Ψ^a e Ψ^b sejam equivalentes se e somente se eles geram o mesmo *ranking* quando aplicados aos mesmos conjuntos de documentos. Neste caso, dois modelos funcionais são equivalentes independente de qualquer consulta funcional qf do conjunto de consultas funcionais de Ψ^a e de Ψ^b.

Representar um modelo na estrutura funcional significa que suas funções de similaridade e forma de representação de documentos e consultas sejam traduzidos em uma mesma linguagem funcional, o λ-cálculo. O objetivo dessa representação é obter o modelo no formalismo da estrutura funcional identificando e caracterizando as funções. A Figura 3.1 mostra um esquema da comparação de equivalência entre os modelos. Dois modelos A e B contendo semânticas diferentes são traduzidos através da estrutura funcional para os modelos funcionais λ-A e λ-B, respectivamente. Assim, ambos estarão representados em uma mesma linguagem (λ-cálculo) e podemos compará-los.

A comparação entre os modelos é importante para reutilização de código ou escolha da implementação de um modelo e melhor entendimento da semântica dos modelos [39].

Neste capítulo apresentamos o meta-modelo funcional baseado no λ-cálculo e suas definições para representação e comparação de equivalência entre modelos de RI. No capítulo seguinte, mostramos aplicações da estrutura funcional através da representação, comparação e combinação de alguns modelos de RI.

Capítulo 4

Uma Proposta de Representação dos Modelos Clássicos de RI

Neste capítulo descrevemos os modelos clássicos de RI. Em seguida é proposta uma representação destes modelos na estrutura funcional. Finalmente, utilizamos a representação na estrutura funcional para identificar a representação equivalente dos modelos clássicos no espaço vetorial. Além disso como consequência das propostas apresentadas, realizamos comparações entre os modelos.

Seja A um modelo clássico de RI qualquer, utilizamos a notação $A \rightarrow \Psi^a$ para denotar que o modelo A é traduzido, através da estrutura funcional, para o modelo funcional Ψ^a. Em seguida, através das funções caracterizadas no modelo funcional Ψ^a, encontramos um modelo vetorial modificado V_a que é traduzido, através da estrutura funcional, para o modelo funcional Ψ^{va} (notação $V_a \rightarrow \Psi^{va}$). Desta forma, podemos fazer comparações entre os modelos funcionais Ψ^a e Ψ^{va} utilizando uma mesma notação.

4.1 Representação $V \rightarrow \Psi^v$

4.1.1 Modelo Vetorial (V)

O modelo vetorial foi inicialmente proposto por Gerard Salton [44]. A apresentação deste modelo segue [3]. Neste modelo, todos os objetos relevantes para o sistema de recuperação de informação são

representados como vetores: termos, documentos e consultas. Uma medida entre vetores é utilizada para ordenar os documentos recuperados para uma consulta.

Cada termo k_i é representado como um vetor t-dimensional, em que t é o número de termos distintos da coleção. No modelo vetorial, o vetor $\vec{k_i}$ representa o termo k_i. Se c_s é o s-ésimo elemento do vetor $\vec{k_i}$, então $\vec{k_i} = (c_1, c_2, \ldots, c_t)$ onde

$$\begin{cases} c_s = 0 \Leftrightarrow s \neq i \\ c_s = 1 \Leftrightarrow s = i \end{cases}$$

ou seja,

$$\begin{aligned} \vec{k_1} &= (1, 0, 0, \ldots, 0) \\ \vec{k_2} &= (0, 1, 0, \ldots, 0) \\ &\vdots \\ \vec{k_t} &= (0, 0, 0, \ldots, 1) \end{aligned}$$

O conjunto de todos os vetores de termos **K** = $\{\vec{k_1}, \vec{k_2}, \ldots, \vec{k_t}\}$ é linearmente independente e forma a base canônica para o espaço $\mathbb{R}^t$ do modelo vetorial. Os vetores de termos são todos ortogonais entre si e, como conseqüência, os termos correspondentes são considerados independentes. A presença de um termo não indica a presença ou ausência de outro termo. A presença de um termo também não aponta sua relação com os outros termos.

Vetores de documentos e consultas são representados utilizando o conjunto de vetores de termos **K**. Estes vetores são construídos como uma combinação linear dos vetores de termos. O vetor $\vec{d_j}$ associado ao documento d_j é definido por:

$$\vec{d_j} = \sum_{i=1}^{t} w_{i,j}\vec{k_i} \quad ou \quad \vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$$

Analogamente, o vetor para a consulta q é definido por:

$$\vec{q} = \sum_{i=1}^{t} w_{i,q}\vec{k_i} \quad ou \quad \vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{t,q}).$$

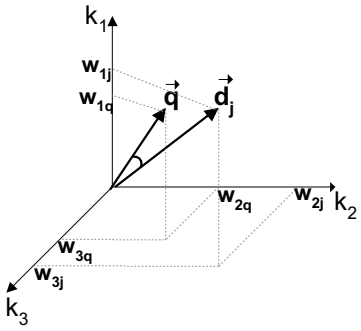


Fig. 4.1: Exemplo dos vetores documento e consulta no espaço vetorial tridimensional ($t = 3$)

Nas igualdades acima, $w_{i,j}$ e $w_{i,q}$ são pesos do termo k_i no documento d_j e na consulta q, respectivamente. A representação gráfica dos vetores de documentos e consultas é exemplificada na Figura 4.1.

A definição de peso dos termos no modelo vetorial é baseada em estatísticas de ocorrência no documento e na coleção.

Um esquema comum para o primeiro fator é empregar a freqüência do termo k_i no documento d_j. O fator de freqüência do termo é usualmente referenciado por tf e fornece uma medida de quão bom é o termo para descrever o conteúdo do documento [45].

A freqüência inversa do documento está relacionada à importância de um termo na coleção de documentos. Em geral, esse fator é indicado por idf e assume que a importância de um termo é inversamente proporcional ao número de documentos no qual ele aparece [45].

A definição dos pesos de termos mais eficiente para a recuperação de informação balanceia esses fatores [3], conforme é mostrado a seguir.

Seja N o número total de documentos no sistema e n_i o número de documentos no qual o termo k_i aparece. Seja $freq_{i,j}$ a freqüência do termo k_i no documento d_j, $max(s)$ uma função que retorna o termo s com maior freqüência na coleção de documentos. A freqüência normalizada $f_{i,j}$ do termo k_i no documento d_j é dada por

$$f_{i,j} = \frac{freq_{i,j}}{max(s) freq_{s,j}}$$

em que o máximo é computado sob todos os termos mencionados no texto do documento d_j. Se o termo k_i não aparece no documento d_j, então $freq_{i,j} = 0$. A freqüência inversa idf_i para k_i é dada por

$$idf_i = log\frac{N}{n_i}$$

ou por variações desta fórmula. O peso do termo k_i no documento d_j é dado por

$$w_{i,j} = f_{i,j} \cdot idf_i$$

Essa estratégia de peso dos termos é denominada tf-idf [3].

O modelo vetorial avalia o grau de similaridade do documento d_j em relação à consulta q como a correlação entre os vetores $\vec{d_j}$ e $\vec{q}$. A relevância de um documento para uma consulta é proporcional à distância entre os respectivos vetores. Usualmente, essa correlação é quantificada pelo co-seno do ângulo entre esses dois vetores. Isto é,

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

.

Observa-se que a similaridade do documento em relação à consulta baseada no co-seno utiliza um fator de normalização no tamanho do documento. Documentos grandes são mais propensos a serem recuperados que documentos pequenos uma vez que os maiores, comumente têm um conjunto de termos maior que os menores [45].

Como $w_{i,j} \geq 0$ e $w_{i,q} \geq 0$, então $0 \leq sim(d_j, q) \leq 1$. Os documentos mais similares (mais próximo no espaço) à consulta são considerados relevantes para o usuário e retornados como resposta para a consulta. Após o cálculo dos graus de similaridade, é possível montar uma lista ordenada (*ranking*)

de todos os documentos e seus respectivos graus de relevância à consulta, em ordem decrescente de relevância.

O modelo vetorial apresenta algumas vantagens: (i) a atribuição de pesos melhora o desempenho; (ii) sua estratégia de casamento parcial possibilita a recuperação de documentos que aproximam as condições da consulta; (iii) os documentos são ordenados de acordo com seu grau de similaridade com a consulta. A maior desvantagem do modelo vetorial é assumir que os termos são mutuamente independentes. Outra desvantagem é que um documento relevante pode não conter termos da consulta [3].

4.1.2 Representação λ-vetorial (Ψ^v)

Propomos a seguir uma representação do modelo vetorial na estrutura funcional. Para representar um modelo na estrutura funcional, é necessário definir um modelo funcional Ψ que o represente. A representação do modelo vetorial na estrutura funcional é denotada por Ψ^v, onde $\Psi^v = \langle [df_1^v; ...; df_n^v], [qf^v], \Delta^v] \rangle$ e

- $df_j^v = [peso_j]$. Os documentos funcionais são listas contendo apenas o termo funcional $peso_j$. A função $peso_j$, para um termo k_i, define o peso do termo k_i no documento d_j, denotada por $peso_j(k_i)$.

- $qf^v = [peso_q]$. As consultas funcionais são listas contendo apenas o termo funcional $peso_q$. A função $peso_q$, para um termo k_i, define o peso do termo k_i na consulta q, denotada por $peso_q(k_i)$.

- A função similaridade Δ^v é dada por

$$\lambda x \lambda y.sim^v(x, y) \text{ onde}$$

$$sim^v = \lambda f \lambda g. \frac{\sum_{i=1}^{t} f(k_i).g(k_i)}{\sqrt{\sum_{i=1}^{t} f(k_i)^2} \times \sqrt{\sum_{i=1}^{t} g(k_i)^2}}$$

  Sendo assim, temos

$$\begin{aligned}
\Delta^v(df_j^v, qf^v) &= (\lambda x \lambda y.sim^v(x,y))(df_j^v, qf^v) \\
&= sim^v(x,y)\{x \leftarrow df_j^v, y \leftarrow qf^v\} \\
&= sim^v(df_j^v, qf^v) \\
&= sim^v([peso_j], [peso_q]) \\
&= sim^v(peso_j, peso_q) \\
&= \lambda f \lambda g. \frac{\sum_{i=1}^{t} f(k_i).g(k_i)}{\sqrt{\sum_{i=1}^{t} f(k_i)^2} \times \sqrt{\sum_{i=1}^{t} g(k_i)^2}}(peso_j, peso_q) \\
&= \frac{\sum_{i=1}^{t} f(k_i).g(k_i)}{\sqrt{\sum_{i=1}^{t} f(k_i)^2} \times \sqrt{\sum_{i=1}^{t} g(k_i)^2}}\{f \leftarrow peso_j, g \leftarrow peso_q\} \\
&= \frac{\sum_{i=1}^{t} peso_j(k_i).peso_q(k_i)}{\sqrt{\sum_{i=1}^{t} peso_j(k_i)^2} \times \sqrt{\sum_{i=1}^{t} peso_q(k_i)^2}}
\end{aligned}$$

Portanto,

$$\Delta^v(df_j^v, qf^v) = \frac{\sum_{i=1}^{t} peso_j(k_i).peso_q(k_i)}{\sqrt{\sum_{i=1}^{t} peso_j(k_i)^2} \times \sqrt{\sum_{i=1}^{t} peso_q(k_i)^2}}.$$

Note que esta função similaridade satisfaz as propriedades de normalização, reflexividade e simetria. Portanto ela é do tipo Δ_{nrs}. A propriedade de normalização é válida, pois a função cosseno entre dois vetores cujas coordenadas são positivas retorna um valor entre 0 e 1 ($0 \leq cos(\vec{a}, \vec{b}) \leq 1$). A propriedade reflexividade é válida, pois $cos(\vec{a}, \vec{a}) = 1$ e a propriedade da simetria também é válida, pois $cos(\vec{a}, \vec{b}) = cos(\vec{b}, \vec{a})$.

4.2 Representação $P \rightarrow \Psi^p$, $V_p \rightarrow \Psi^{vp}$, $\Psi^p \simeq \Psi^{vp}$

4.2.1 Modelo Probabilístico (P)

O modelo probabilístico clássico foi introduzido em 1976 por Roberston e Sparck Jones [43]. Este modelo trata do problema de RI dentro de um *framework* probabilístico [51]. A apresentação deste modelo segue [3].

O modelo probabilístico é baseado no princípio probabilístico: dado uma consulta q e um documento d_j na coleção, o modelo probabilístico estima a probabilidade do usuário considerar o documento d_j relevante. O modelo probabilístico assume que esta probabilidade de relevância depende

somente das representações da consulta e do documento. Além disso, o modelo assume que há um conjunto resposta ideal de documentos, denominado **R**, que maximiza toda a probabilidade de relevância para o usuário. Documentos no conjunto **R** são considerados relevantes para uma consulta q. Documentos que não estão neste conjunto são considerados não relevantes. Como o conjunto **R** não é conhecido, o modelo se baseia em estimativas conforme discutidas a seguir.

No modelo probabilístico os pesos dos termos são todos binários, isto é, $w_{i,j} \in \{0,1\}$ e $w_{i,q} \in \{0,1\}$. E, de forma análoga ao modelo vetorial, os documentos e consultas, d_j e q, são representados por vetores $\vec{d_j}$ e $\vec{q}$. Seja $\mathbf{R}_q$ o conjunto de documentos que foram estimados como relevantes para a consulta q, ou seja, uma estimativa para o conjunto ideal, e seja $\overline{\mathbf{R}}_q$ o seu complemento. Dado um documento d_j, $P(\mathbf{R}_q|\vec{d_j})$ é a probabilidade de d_j pertencer a $\mathbf{R}_q$. Em outras palavras, $P(\mathbf{R}_q|\vec{d_j})$ é a probabilidade do documento d_j ser relevante para a consulta q. Analogamente, $P(\overline{\mathbf{R}}_q|\vec{d_j})$ é a probabilidade do documento d_j não ser relevante para q. A similaridade entre um documento d_j e uma consulta q é definida por:

$$sim(d_j, q) = \frac{P(\mathbf{R}_q|\vec{d_j})}{P(\overline{\mathbf{R}}_q|\vec{d_j})}$$

Utilizando a regra de Bayes,

$$sim(d_j, q) = \frac{P(\vec{d_j}|\mathbf{R}_q) \times P(\mathbf{R}_q)}{P(\vec{d_j}|\overline{\mathbf{R}}_q) \times P(\overline{\mathbf{R}_q})}$$

onde $P(\vec{d_j}|\mathbf{R}_q)$ é a probabilidade de aleatoriamente selecionar o documento d_j do conjunto $\mathbf{R}_q$ de documentos relevantes e $P(\mathbf{R}_q)$ é a probabilidade que um documento aleatoriamente selecionado de toda a coleção seja relevante. Os significados para $P(\vec{d_j}|\overline{\mathbf{R}}_q)$ e $P(\overline{\mathbf{R}}_q)$ são análogos considerando o conjunto $\overline{\mathbf{R}}_q$.

Para simplificar, o modelo probabilístico assume independência de termos. Além disso, para calcular as probabilidades, o modelo aplica a elas uma série de transformações preservando a ordem com o objetivo de obter uma estimativa numérica para o *ranking* do documento d_j baseada em termos.

Tais transformações incluem a aplicação de logaritmos, regras probabilísticas e a transformação de probabilidades que levam em conta o documento d_j para probabilidades que consideram pesos de termos em d_j. Os detalhes são descritos em [51]. Como resultado destas transformações, temos:

$$sim(d_j, q) = \sum_{i=1}^{t} w_{i,j} \times w_{i,q} \times \left(log \frac{P(k_i|\mathbf{R}_q)}{1 - P(k_i|\mathbf{R}_q)} + log \frac{1 - P(k_i|\overline{\mathbf{R}}_q)}{P(k_i|\overline{\mathbf{R}}_q)} \right)$$

onde $P(k_i|\mathbf{R}_q)$ é a probabilidade do termo k_i está presente em um documento aleatoriamente selecionado do conjunto $\mathbf{R}_q$ e $P(k_i|\overline{\mathbf{R}}_q)$ é a probabilidade do termo k_i não estar presente em um documento aleatoriamente selecionado do conjunto $\mathbf{R}_q$. Esta é a expressão clássica para determinar o *ranking* no modelo probabilístico.

Visto que o conjunto $\mathbf{R}_q$ não é conhecido inicialmente, é necessário dispor de um método para inicialmente computar as probabilidades $P(k_i|\mathbf{R}_q)$ e $P(k_i|\overline{\mathbf{R}}_q)$. Existem várias alternativas para tal computação [3]. Dado esta estimativa inicial, pode-se então recuperar os documentos e fornecer um *ranking* probabilístico inicial para a consulta. Em seguida, este *ranking* pode ser melhorado através de uma interação com o usuário. Este processo pode ser repetido recursivamente. Algumas maneiras de estimar e melhorar o *ranking* probabilístico podem ser encontradas em [3].

A principal vantagem do modelo probabilístico é o fato dos documentos serem ordenados em ordem decrescente de acordo com a probabilidade de serem relevantes. As principais desvantagens deste modelo são o fato de que, para várias aplicações, a distribuição dos termos entre documentos relevantes e irrelevantes não estará disponível, o fato de que o método não leva em conta a frequência com que os termos ocorrem dentro dos documentos [3].

4.2.2 Modelo λ-probabilístico (Ψ^p)

Propomos a seguir uma representação do modelo probabilístico na estrutura funcional. A representação do modelo probabilístico na estrutura funcional é denotada por Ψ^p, onde

$$\Psi^p = \langle [df_1^p, ..., df_n^p], [qf^p], \Delta^p \rangle$$

é um modelo funcional tal que,

- $df_j^p = [peso_j]$. Os documentos funcionais são listas contendo apenas o termo funcional $peso_j$. A função $peso_j$ define o peso $peso_j(k_i)$ no modelo probabilístico, ou seja, o peso do termo k_i no documento d_j.

- $qf^p = [peso_q,\ prob_q,\ \overline{prob}_q]$. As consultas funcionais são listas contendo os termos funcionais $peso_q$, $prob_q$ e $\overline{prob}_q$.

  O termo funcional $peso_q$ para um termo k_i define o peso do termo k_i na consulta q, denotado por $peso_q(k_i)$.

  O termo funcional $prob_q$ é denotado por

  $$\lambda x.P(x|\mathbf{R}_q),$$

  onde $\mathbf{R}_q$ é uma lista de documentos que foram estimados como relevantes para a consulta q e P é a função probabilidade. Sendo assim, temos que $prob_q(k_i) = P(k_i|\mathbf{R}_q)$, ou seja, é a probabilidade do termo k_i estar presente em um documento escolhido aleatoriamente do conjunto $\mathbf{R}_q$.

  O termo funcional $\overline{prob}_q$ é denotado por

  $$\lambda x.P(x|\overline{\mathbf{R}}_q)$$

  onde $\overline{\mathbf{R}}_q$ é o complemento de $\mathbf{R}_q$, ou seja, a lista de documentos que não foram estimados como relevantes para a consulta q e P é a função probabilidade. Sendo assim, temos que $\overline{prob}_q(k_i) = P(k_i|\overline{\mathbf{R}}_q)$, ou seja, é a probabilidade do termo k_i estar presente em um documento escolhido aleatoriamente do conjunto $\overline{\mathbf{R}}_q$.

- Função similaridade Δ^p é dada por

  $$\lambda x \lambda y.sim^p(x, y)$$

onde

$$sim^p = \lambda f \lambda g \lambda h \lambda h_1 . \sum_{i=1}^{t} f(k_i) \times g(k_i) \times \left(\log \frac{h(k_i)}{1 - h(k_i)} + \log \frac{1 - h_1(k_i)}{h_1(k_i)} \right).$$

Sendo assim, temos

$$\begin{aligned}
&\Delta^p(df_j^p, qf^p) = (\lambda x \lambda y . sim^p(x,y))(df_j^p, qf^p) \\
&= sim^p(x,y)\{x \leftarrow df_j^p, y \leftarrow qf^p\} \\
&= sim^p(df_j^p, qf^p) \\
&= sim^p([peso_j], [peso_q, prob_q, \overline{prob_q}]) \\
&= sim^p(peso_j, peso_q, prob_q, \overline{prob_q}) \\
&= (\lambda f \lambda g \lambda h \lambda h_1 . \textstyle\sum_{i=1}^{t} f(k_i) \times g(k_i) \times (\log \frac{h(k_i)}{1-h(k_i)} + \log \frac{1-h_1(k_i)}{h_1(k_i)}))\ \ (peso_j, peso_q, prob_q, \overline{prob_q}) \\
&= \textstyle\sum_{i=1}^{t} f(k_i) \times g(k_i) \times (\log \frac{h(k_i)}{1-h(k_i)} + \log \frac{1-h_1(k_i)}{h_1(k_i)}) \\
&\{f \leftarrow peso_j, g \leftarrow peso_q, h \leftarrow prob_q, h_1 \leftarrow \overline{prob_q})\} \\
&= \textstyle\sum_{i=1}^{t} peso_j(k_i) \times peso_q(k_i) \times (\log \frac{h(k_i)}{1-h(k_i)} + \log \frac{1-h_1(k_i)}{h_1(k_i)}) \\
&= \textstyle\sum_{i=1}^{t} peso_j(k_i) \times peso_q(k_i) \times (\log \frac{P(k_i|\mathbf{R}_q)}{1-P(k_i|\mathbf{R}_q)} + \log \frac{1-P(k_i|\overline{\mathbf{R}}_q)}{P(k_i|\overline{\mathbf{R}}_q)})
\end{aligned}$$

Portanto,

$$\Delta^p(df_j^p, qf^p) = \sum_{i=1}^{t} peso_j(k_i) \times peso_q(k_i) \times \left(\log \frac{P(k_i|\mathbf{R}_q)}{1 - P(k_i|\mathbf{R}_q)} + \log \frac{1 - P(k_i|\overline{\mathbf{R}}_q)}{P(k_i|\overline{\mathbf{R}}_q)} \right)$$

Podemos definir uma constante de normalização η, de tal forma que a função Ψ^p retorne um valor entre 0 e 1 ($0 \leq \Delta^p(df_j^p, qf^p) \leq 1$) e que seu maior valor (1) ocorra quando df_j^p for igual a qf^p. Desta forma, as propriedades de normalização e reflexividade são válidas. Por outro lado, a propriedade de simetria não é válida, pois podemos ter $\Delta^p(df_j^p, qf^p) \neq \Delta^p(qf^p, df_j^p)$. Portanto, a função similaridade é do tipo Δ_{nr}, considerando a constante de normalização.

4.2.3 Modelo vetorial-probabilístico (V_p)

Propomos a seguir um modelo vetorial estendido equivalente ao modelo Ψ^p descrito anteriormente. No modelo clássico vetorial, o conjunto de termos $\{\vec{k}_1, \vec{k}_2, \ldots, \vec{k}_t\}$ forma os eixos do espaço vetorial. Os documentos e consultas são representados como vetores no espaço: $\vec{d}_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$ e

$\vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{t,q})$ e a equação da função similaridade é dada pelo cosseno formado entre estes dois vetores (veja seção 4.1.1).

Propomos neste trabalho um modelo vetorial estendido que utiliza o termo funcional δ_q onde

$$\delta_q = \lambda y. \left(\log \frac{P(y|\mathbf{R}_q)}{1 - P(y|\mathbf{R}_q)} + \log \frac{1 - P(y|\overline{\mathbf{R}_q})}{P(y|\overline{\mathbf{R}_q})} \right)$$

Além disso, a pesagem dos termos dos vetores documento e consulta e o cálculo da similaridade também são modificados, conforme descrito a seguir.

O termo funcional δ_q é uma função que tem como domínio o conjunto dos termos **K** onde $\mathbf{K} = \{k_1, k_2, \ldots, k_t\}$ e contradomínio o conjunto $\mathbb{R}$. Para cada i ($1 \leq i \leq t$) temos que $\delta_q(k_i) \in \mathbb{R}$.

Considere uma matriz diagonal M tal que:

$$M = \begin{bmatrix} \delta_q(k_1) & 0 & \cdots & 0 \\ 0 & \delta_q(k_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \delta_q(k_t) \end{bmatrix}$$

Os valores presentes na matriz M servem como um fator discriminante para a consulta q. Sendo assim, podemos então descrever o modelo vetorial modificado, da seguinte forma:

- O documento d_j é representado no espaço vetorial pelo vetor $\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$ tal que $w_{i,j} \in \{0, 1\}$
- Temos agora um novo vetor consulta $\vec{q}_{mod}$ no espaço vetorial dado pela combinação linear entre o vetor consulta $\vec{q}$ e a matriz M, ou seja, $\vec{q}_{mod} = \vec{q} \times M$.

  Seja o vetor consulta $\vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{t,q})$ tal que $w_{i,q} \in \{0, 1\}$ e a matriz diagonal M descrita acima. Temos então que:

$$\vec{q}_{mod} = (w_{1,q}, w_{2,q}, \ldots, w_{t,q}) \times \begin{bmatrix} \delta_q(k_1) & 0 & \cdots & 0 \\ 0 & \delta_q(k_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \delta_q(k_t) \end{bmatrix}$$

Portanto, desenvolvendo esta operação temos que o vetor consulta $\vec{q}_{mod}$ é dado por:

$$(w_{1,q} \times \delta_q(k_1), w_{2,q} \times \delta_q(k_2), \ldots, w_{t,q} \times \delta_q(k_t))$$

- A equação da função de similaridade é dada por:

$$\begin{aligned} sim(d_j, q) &= \vec{d_j} \bullet \vec{q}_{mod} \\ &= \sum_{i=1}^{t} w_{i,j} \times w_{i,q} \times \delta_q(i), \end{aligned}$$

ou seja, a similaridade é dada pelo produto interno entre o vetor documento $\vec{d_j}$ e o vetor consulta modificado $\vec{q}_{mod}$.

A Figura 4.2 mostra esse modelo vetorial estendido.

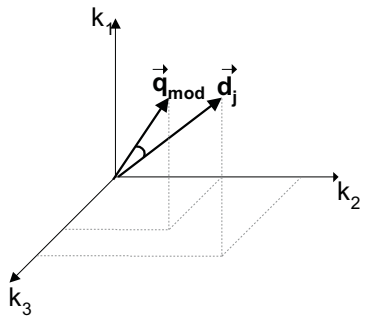


Fig. 4.2: Exemplo do modelo vetorial-probabilístico no espaço vetorial

4.2.4 Modelo λ-vetorial-probabilístico (Ψ^{vp})

Agora representamos o modelo vetorial-probabilístico (V_p) na estrutura funcional e mostramos sua equivalência com o modelo probabilístico. Para representar o modelo vetorial modificado na estrutura funcional, definimos o modelo funcional $\Psi^{vp} = \langle [df_1^{vp}; ...; df_n^{vp}], [qf^{vp}], \Delta^{vp}] \rangle$, onde:

- $df_j^{vp} = [peso_j]$. Os documentos funcionais são listas contendo apenas o termo funcional $peso_j$. A função $peso_j$ para um termo k_i define o peso do termo k_i no documento d_j, denotada por $peso_j(k_i)$.

- $qf^{vp} = [peso_q, \delta_q]$. As consultas funcionais são listas contendo o termo funcional $peso_q$ e δ_q.

  A função $peso_q$ para um termo k_i define o peso do termo k_i na consulta q, denotada por $peso_q(k_i)$.

  A função $\delta_q(k_i)$ é um fator discriminante do termo k_i em relação à consulta q.

- A função similaridade Δ^{vp} é dada por

$$\lambda x \lambda y.sim^{vp}(x, y)$$

onde

$$sim^{vp} = \lambda f \lambda g \lambda h. \sum_{i=1}^{t} f(k_i) \times g(k_i) \times h(k_i).$$

Sendo assim, temos

$$\begin{aligned}
\Delta^{vp}(df_j^{vp}, qf^{vp}) &= (\lambda x \lambda y.sim^{vp}(x, y))(df_j^{vp}, qf^{vp}) \\
&= sim^{vp}(x, y)\{x \leftarrow df_j^{vp}, y \leftarrow qf^{vp}\} \\
&= sim^{vp}(df_j^{vp}, qf^{vp}) \\
&= sim^{vp}([peso_j], [peso_q, \delta_q]) \\
&= sim^{vp}(peso_j, peso_q, \delta_q) \\
&= (\lambda f \lambda g \lambda h. \textstyle\sum_{i=1}^{t} f(k_i) \times g(k_i) \times h(k_i))\ \ (peso_j, peso_q, \delta_q) \\
&= \textstyle\sum_{i=1}^{t} f(k_i) \times g(k_i) \times h(k_i)\{f \leftarrow peso_j, g \leftarrow peso_q, h \leftarrow \delta_q\} \\
&= \textstyle\sum_{i=1}^{t} peso_j(k_i) \times peso_q(k_i) \times \delta_q(k_i)
\end{aligned}$$

Portanto,

$$\Delta^{vp}(df_j^{vp}, qf^{vp}) = \sum_{i=1}^{t} peso_j(k_i) \times peso_q(k_i) \times \delta_q(k_i).$$

Substituindo $\delta_q(k_i)$, conforme definição, temos:

$$\Delta^{vp}(df_j^{vp}, qf^{vp}) = \sum_{i=1}^{t} peso_j(k_i) \times peso_q(k_i) \times \left(\log \frac{P(k_i|\mathbf{R}_q)}{1 - P(k_i|\mathbf{R}_q)} + \log \frac{1 - P(k_i|\overline{\mathbf{R}_q})}{P(k_i|\overline{\mathbf{R}_q})} \right).$$

Logo, $\Delta^{vp}(df_j^{vp}, qf^{vp}) = \Delta^{p}(df_j^{p}, qf^{p})$.

Verificamos que os modelos funcionais correspondentes Ψ^p e Ψ^{vp} são equivalentes por construção. Existe a função bijetora identidade $\phi : \{df_1^p, \ldots, df_n^p\} \rightarrow \{df_1^{vp}, \ldots, df_n^{vp}\}$ que mapeia documentos de Ψ^p em Ψ^{vp}. Além disso, as duas propriedades de equivalência são satisfeitas pois possuem a mesma função de similaridade. Portanto, os modelos Ψ^p e Ψ^{vp} são equivalentes e geram o mesmo *ranking*.

4.3 Representação $B \rightarrow \Psi^b$, $V_b \rightarrow \Psi^{vb}$, $\Psi^b \simeq \Psi^{vb}$

4.3.1 Modelo Booleano (B)

O modelo booleano é um modelo de recuperação de informação baseado na teoria dos conjuntos e álgebra booleana. Este foi o primeiro modelo usado em RI e o mais utilizado até meados da década de 1990 em vários sistemas comerciais bibliográficos [3]. A apresentação deste modelo segue [3].

A estratégia de recuperação é baseada num critério de decisão binário (isto é, um documento é considerado relevante ou irrelevante para uma consulta) sem qualquer noção de casamento parcial. O fato de o modelo booleano não possibilitar a ordenação dos resultados por ordem de relevância é uma de suas principais desvantagens, já que esta classificação é uma característica considerada essencial em muitos dos sistemas de RI modernos.

Este modelo considera uma consulta como uma expressão booleana convencional, que liga seus termos através de conectivos lógicos AND, OR e NOT. Suponha, por exemplo, que os termos do vocabulário sejam k_1, k_2, k_3 e k_4. A expressão booleana q a seguir é uma consulta no modelo booleano: $q = (\neg k_1 \wedge k_2 \wedge k_4) \vee k_3$. Da mesma forma, um documento também é representado por uma expressão booleana, que expressa os termos que ele contém e que ele não contém. O documento d_j, tal que, $d_j = \neg k_1 \wedge k_2 \wedge \neg k_3 \wedge k_4$ representa o documento que contém os termos k_2 e k_4 e não contém os termos k_1 e k_3. A similaridade é dada por:

$$sim(d_j, q) = \begin{cases} 1, & \text{se } d_j \text{ implica } q \\ 0, & caso \text{ contrário} \end{cases}$$

onde a implicação das expressões booleanas corresponde ao conceito de implicação lógica [49], descrito a seguir.

Definição 4.1 (valoração). *Valoração é uma função binária* $V : \boldsymbol{E} \rightarrow \{0, 1\}$ *cujo domínio é constituído pelo conjunto das expressões booleanas (formadas pelos termos e os conectivos lógicos* $\wedge$,$\vee$ *e* $\neg$*) e o contradomínio é o conjunto que contém apenas os valores 0 e 1. Além disso, temos as seguintes propriedades:*

- *Se* $V[k_i] = 1$*, então* $V[\neg k_i] = 0$*.*
- *Se* $V[\neg k_i] = 0$*, então* $V[k_i] = 1$*.*
- $V[k_i \vee k_j] = 1$ *se e somente se* $V[k_i] = 1$ *e/ou* $V[k_j] = 1$*.*
- $V[k_i \wedge k_j] = 1$ *se e somente se* $V[k_i] = 1$ *e* $V[k_j] = 1$*.*

Definição 4.2 (implicação). *Dadas duas expressões booleanas* E_1 *e* E_2*,* E_1 *implica* E_2 *se e somente se para toda valoração V,*

$$\textit{se } V[E_1] = 1 \textit{ então } V[E_2] = 1.$$

Exemplo: O documento $d_j = \neg k_1 \wedge k_2 \wedge \neg k_3 \wedge k_4$ implica a consulta $q = (\neg k_1 \wedge k_2 \wedge k_4) \vee k_3$, pois temos que:

$$d_j \text{ implica } q \Leftrightarrow \forall \text{ valoração } V, \text{ se } V[d_j] = 1 \text{ então } V[q] = 1.$$

Mas, $V[\neg k_1 \wedge k_2 \wedge \neg k_3 \wedge k_4] = 1$ se e somente se $V[k_1] = 0$ e $V[k_2] = 1$ e $V[k_3] = 0$ e $V[k_4] = 1$. Tendo em vista estas valorações, temos que $V[(\neg k_1 \wedge k_2 \wedge k_4) \vee k_3] = 1$, pois $V[\neg k_1 \wedge k_2 \wedge k_4] = 1$. Logo, temos que d_j implica q.

As principais desvantagens do modelo booleano são: (i) dificuldade para o usuário traduzir sua necessidade utilizando os conectivos lógicos; (ii) não apresenta a noção de casamento parcial. As vantagens do modelo booleano são a facilidade de implementação e uma semântica precisa das expressões booleanas. Apesar dos problemas deste modelo, dada sua simplicidade e seu formalismo, recebeu uma enorme atenção há alguns anos atrás e foi adotado por muitos sistemas bibliográficos comerciais. Além disso, existem variações deste modelo que implementam o conceito de *ranking*, como os modelos fuzzy e booleano estendido [3, 6].

4.3.2 Modelo λ-booleano (Ψ^b)

Propomos a seguir uma representação do modelo booleano na estrutura funcional. A representação do modelo booleano na estrutura funcional é denotada por Ψ^b, onde

$$\Psi^b = \langle [df_1^b, ..., df_n^b], [qf^b], \Delta^b \rangle$$

é um modelo funcional e,

- $df_j^b = [bol_j]$. Os documentos funcionais são listas contendo apenas o termo funcional bol_j. O termo funcional bol_j representa a função booleana da conjunção de termos pertencentes ao documento d_j e a negação dos termos que não pertencem ao documento d_j, isto é,

$$bol_j = \lambda(x_1, ..., x_t).\bigwedge_{\forall x_i; peso_j(x_i)=1} x_i \bigwedge_{\forall x_i; peso_j(x_i)=0} \neg x_i$$

  Exemplo: Considere os termos do vocabulário k_1, k_2, k_3 e k_4, tais que, $peso_j(k_1) = 0$, $peso_j(k_2) = 1$,$peso_j(k_3) = 0$ e $peso_j(k_4) = 1$. Neste caso,

$$bol_j(k_1, k_2, k_3, k_4) = (\lambda(x_1, x_2, x_3, x_4).\bigwedge_{\forall x_i; peso_j(x_i)=1} x_i \bigwedge_{\forall x_i; peso_j(x_i)=0} \neg x_i)\ (k_1, k_2, k_3, k_4)$$
$$= \bigwedge_{\forall x_i; peso_j(x_i)=1} x_i \bigwedge_{\forall x_i; peso_j(x_i)=0} \neg x_i\ \{x_1 \leftarrow k_1, x_2 \leftarrow k_2, x_3 \leftarrow k_3, x_4 \leftarrow k_4\}$$
$$= (\neg k_1) \wedge (k_2) \wedge (\neg k_3) \wedge (k_4)$$

  Portanto, $bol_j(k_1, k_2, k_3, k_4) = (\neg k_1) \wedge (k_2) \wedge (\neg k_3) \wedge (k_4)$.

- $qf_b = [bol_q]$ As consultas funcionais são listas contendo apenas o termo funcional bol_q. O termo funcional bol_q retorna uma expressão booleana associada a consulta q. Podemos ter, por exemplo, $bol_q(k_1, k_2, k_3, k_4) = (\neg k_1 \wedge k_2 \wedge k_4) \vee k_3$.

- A função similaridade Δ^b é dada por

$$\lambda x.\lambda y.sim^b(x, y) \text{ onde}$$

$$sim^b = \lambda f \lambda g.((f(k_1, \ldots, k_t)\ implica\ g(k_1, \ldots, k_t)) \rightarrow 1 \mid 0)$$

  Sendo assim, temos

$$
\begin{aligned}
\Delta^b(df_j^b, qf^b) &= (\lambda x.\lambda y.sim^b(x,y))\ (df_j^b, qf^b) \\
&= sim^b(x,y)\{x \leftarrow df_j^b, y \leftarrow qf^b\} \\
&= sim^b(df_j^b, qf^b) \\
&= sim^b([bol_j], [bol_q]) \\
&= sim^b(bol_j, bol_q) \\
&= \lambda f \lambda g.((f(k_1,\ldots,k_t)\ implica\ g(k_1,\ldots,k_t)) \rightarrow 1 \mid 0)\ (bol_j, bol_q) \\
&= ((f(k_1,\ldots,k_t)\ implica\ g(k_1,\ldots,k_t)) \rightarrow 1 \mid 0)\ \{f \leftarrow bol_j, g \leftarrow bol_q\} \\
&= (bol_j(k_1,\ldots,k_t)\ implica\ bol_q(k_1,\ldots,k_t)) \rightarrow 1 \mid 0
\end{aligned}
$$

Portanto,

$$
\Delta^b(df_j^b, qf^b) = \begin{cases} 1,\ se\ bol_j(k_1,\ldots,k_t)\ implica\ bol_q(k_1,\ldots,k_t) \\ 0,\ caso\ contrario \end{cases}
$$

Note que esta função similaridade satisfaz as propriedades de normalização e reflexividade, mas não é simétrica. A propriedade de normalização é válida, pois a função de similaridade possui apenas valores 0 e 1. A propriedade de reflexividade é válida, pois o documento funcional df_j^b implica ele mesmo, isto é, $\Delta^b(df_j^b, df_j^b) = 1$. A propriedade de simetria não é válida, pois a implicação não é simétrica. Portanto, a similaridade é do tipo Δ_{nr}.

4.3.3 Modelo vetorial-booleano (V_b)

Descrevemos a seguir uma proposta de como o modelo booleano pode ser representado no espaço vetorial.

O modelo booleano considera que os termos estão ausentes ou presentes em um documento, ou seja, $w_{i,j} = \{0, 1\}$. Uma consulta q é composta de termos ligados por conectivos lógicos, ou seja, é uma expressão booleana.

Podemos representar o modelo booleano no espaço vetorial da seguinte forma:

- O documento d_j é representado pelo vetor $\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$, onde $w_{1,j} \in \{0, 1\}$. Suponha por exemplo $d_j = \neg k_1 \wedge k_2 \wedge \neg k_3 \wedge k_4$. Neste caso, $\vec{d_j} = (0, 1, 0, 1)$.

- No modelo booleano uma consulta q é representada por uma expressão booleana. Por exemplo, a expressão $q = (\neg k_1 \wedge k_2 \wedge k_4) \vee k_3$ é uma consulta no modelo booleano. Esta consulta também pode ser representada como uma expressão na Forma Normal Disjuntiva (FND) completa, definida a seguir.

  Definição 4.3 (Forma Normal Disjuntiva completa). *Uma fórmula booleana está na FND completa se é uma disjunção de conjunção completa de termos.*
  A fórmula $(\neg k_1 \wedge k_2 \wedge k_4) \vee k_3$ é um exemplo de fórmula na FND onde k_1, k_2, k_3, k_4 são termos da coleção. Observe que neste caso, cada subexpressão da disjunção não necessariamente contém todos os termos da coleção. Uma fórmula booleana é uma conjunção completa de termos se ela contém todos os termos da coleção. Neste caso, se k_1, k_2, k_3, k_4 são todos os termos da coleção, então $\neg k_1 \wedge k_2 \wedge k_3 \wedge k_4$ é uma conjunção completa.

  A Forma Normal Disjuntiva completa equivalente a consulta q descrita acima é dada por:

$$\begin{aligned} q_{fndc} &= (\neg k_1 \wedge k_2 \wedge k_4 \wedge \neg k_3) \vee (\neg k_1 \wedge k_2 \wedge k_4 \wedge k_3) \vee (k_1 \wedge \neg k_2 \wedge k_4 \wedge k_3) \\ &\vee (\neg k_1 \wedge \neg k_2 \wedge k_4 \wedge k_3) \vee (k_1 \wedge k_2 \wedge k_4 \wedge k_3) \vee (\neg k_1 \wedge k_2 \wedge \neg k_4 \wedge k_3) \\ &\vee (k_1 \wedge \neg k_2 \wedge \neg k_4 \wedge \neg k_3) \vee (\neg k_1 \wedge \neg k_2 \wedge \neg k_4 \wedge k_3) \vee (k_1 \wedge k_2 \wedge \neg k_4 \wedge k_3). \end{aligned}$$

  A consulta q_{fndc} pode ser representada por um conjunto de vetores onde cada vetor representa uma das conjunções completas da expressão q_{fndc}. Isto significa que q_{fndc} é igual ao conjunto dos vetores que representam as expressões conjuntivas da FND completa. Desta forma, tendo em vista a consulta q_{fndc} temos que $q_{fndc} = \{(0,1,0,1); (0,1,1,1); (1,0,1,1); (0,0,1,1); (1,1,1,1); (0,1,1,0); (1,0,0,0); (0,0,1,0); (1,1,1,0)\}$ onde cada conjunção formada pela tupla (k_1, k_2, k_3, k_4) é um vetor.

- Se $\vec{d_j} = (w_{1,j}, \ldots, w_{t,j})$ e $q_{fndc} = \{\vec{v}_1, \ldots, \vec{v}_u\}$ então a equação da similaridade é dada por:

$$sim(d_j, q_{fndc}) = \begin{cases} 1, & \text{se } \vec{d_j} = \vec{v}_s \text{ para algum } \vec{v}_s \text{ tal que } \vec{v}_s \in q_{fndc} \\ 0, & caso \text{ contrário} \end{cases}$$

4.3.4 Modelo λ-vetorial-booleano (Ψ^{vb})

Propomos a seguir uma representação do modelo vetorial-booleano na estrutura funcional. A representação deste modelo na estrutura funcional é denotada por Ψ^{vb}, onde

$\Psi^{vb} = \langle [df_1^{vb}, ..., df_n^{vb}], [qf^{vb}], \Delta^{vb} \rangle$ é um modelo funcional e,

- $df_j^{vb} = [bol_j]$. Os documentos funcionais são listas contendo apenas o termo funcional bol_j. A função bol_j está definida na seção 4.3.2.

- $qf^{vb} = [bol_q^1, \ldots, bol_q^u]$ As consultas funcionais são listas contendo os termos funcionais $bol_q^1, \ldots, bol_q^u$ onde bol_q^s para $1 \leq s \leq u$ é a função que define a s-ésima expressão booleana conjuntiva completa da FND equivalente a consulta. O termo funcional bol_q^s é definido por:

$$bol_q^s = \lambda(x_1, ..., x_t).\bigwedge_{\forall x_i; peso_q^s(x_i)=1} x_i \bigwedge_{\forall x_i; peso_q^s(x_i)=0} \neg x_i$$

- A função similaridade é dada por

$$\Delta^{vb}(df_j^{vb}, qf^{vb}) \begin{cases} 1, & \text{se } bol_j(k_1, \ldots, k_t) = bol_q^s(k_1, \ldots, k_t) \text{ para algum } s \text{ onde } 1 \leq s \leq u \\ 0, & caso \text{ contrário} \end{cases}$$

Em seguida, comparamos os modelos booleano e vetorial-booleano mostrando que eles são equivalentes.

Sejam $\Psi_b = \langle [df_1^b, ..., df_n^b], [qf^v], \Delta^b \rangle$ e $\Psi^{vb} = \langle [df_1^{vb}, ..., df_n^{vb}], [qf^{vb}], \Delta^{vb} \rangle$ os modelos funcionais booleano e vetorial-booleano, respectivamente conforme apresentamos anteriormente. A representação de um documento funcional df_j^b no modelo funcional booleano é idêntica à representação de um documento funcional no modelo vetorial-booleano. Logo a função identidade $\phi_1 : \{df_1^b, \ldots, df_n^b\} \rightarrow \{df_1^{vb}, ..., df_n^{vb}\}$ mapeia os documentos de Ψ^b em Ψ^{vb}. Esta função é a função identidade $\lambda x.x$. Mostramos a seguir que para toda consulta qf, os documentos retornados são iguais nos dois modelos, ou seja, as duas propriedades de equivalência (veja Definição 3.11) são satisfeitas.

Como nos modelos booleano e vetorial-booleano o valor da similaridade é 0 ou 1, basta provarmos que $\Delta^b(qf, df_j^b) = 1 \Leftrightarrow \Delta^{vb}(qf, df_j^{vb}) = 1$ para que as duas propriedades sejam satisfeitas. Portanto devemos provar que:

$$bol_j(k_1, \ldots, k_t) \text{ implica } bol_q(k_1, \ldots, k_t)$$
$$\Updownarrow$$
$$bol_j(k_1, \ldots, k_t) = bol_q^s(k_1, \ldots, k_t) \text{ para algum } s;\ (1 \leq s \leq u).$$

($\Rightarrow$) Primeiro, provamos a implicação ($\Rightarrow$), ou seja,

$$bol_j(k_1, \ldots, k_t) \text{ implica } bol_q(k_1, \ldots, k_t)$$
$$\Downarrow$$
$$bol_j(k_1, \ldots, k_t) = bol_q^s(k_1, \ldots, k_t) \text{ para algum } s;\ (1 \leq s \leq u).$$

Temos que,

$$bol_j(k_1, \ldots, k_t) \text{ implica } bol_q(k_1, \ldots, k_t)$$
$$\Updownarrow$$
$$\forall \text{ valoração } V, \text{se } V[bol_j(k_1, \ldots, k_t)] = 1 \text{ então } V[bol_q(k_1, \ldots, k_t)] = 1.$$

Suponha por absurdo, que $\forall\ s$, $V[bol_j(k_1, \ldots, k_t)] \neq V[bol_q^s(k_1, \ldots, k_t)]$. Logo, $\forall\ s$ temos que $\exists\ k_i\ (1 \leq i \leq t)$ tal que k_i^s e $\neg k_i^s$ ocorrem em $bol_j(k_1, \ldots, k_t)$ e $bol_q^s(k_1, \ldots, k_t)$, respectivamente, ou vice-versa.

Temos então, dois casos possíveis:

1^o Caso - k_i^s ocorre em $bol_j(k_1, \ldots, k_t)$

$\neg k_i^s$ ocorre em $bol_q^s(k_1, \ldots, k_t)$

2^o Caso - $\neg k_i^s$ ocorre em $bol_j(k_1, \ldots, k_t)$

k_i^s ocorre em $bol_q^s(k_1, \ldots, k_t)$

Se ocorrer o 1^o Caso, defina uma valoração V tal que $V[k_i^s] = 1$. Se ocorrer 2^o Caso, defina uma valoração V tal que $V[k_i^s] = 0$.

Repetindo este raciocínio $\forall\ s$, definimos uma valoração para um sub-conjunto dos termos que ocorrem em $bol_j(k_1, \ldots, k_t)$. Para o resto dos termos de bol_j, suponha $\{k_{r1}, \ldots, k_{rx}\}$, faça $V[k_{ri}] = 1$ para todo i. Para esta valoração, temos $V[bol_j(k_1, \ldots, k_t)] = 1$ e $V[bol_q^s(k_1, \ldots, k_t)] = 0$ para todo s. Logo, $V[bol_q(k_1, \ldots, k_t)] = 0$. Isto é um absurdo, pois temos que:

$$\forall \text{ valoração } V, \text{se } V[bol_j(k_1, \ldots, k_t)] = 1 \text{ então } V[bol_q(k_1, \ldots, k_t)] = 1.$$

Portanto, é válido que:

$$bol_j(k_1, \ldots, k_t) \text{ implica } bol_q(k_1, \ldots, k_t)$$
$$\Downarrow$$
$$bol_j(k_1, \ldots, k_t) = bol_q^s(k_1, \ldots, k_t) \text{ para algum } s;\ (1 \leq s \leq u).$$

($\Leftarrow$) Agora, demonstramos que:

$$bol_j(k_1, \ldots, k_t) = bol_q^s(k_1, \ldots, k_t) \text{ para algum } s;\ (1 \leq s \leq u)$$
$$\Downarrow$$
$$bol_j(k_1, \ldots, k_t) \text{ implica } bol_q(k_1, \ldots, k_t).$$

Suponha que $V[bol_j(k_1, \ldots, k_t)] = V[bol_q^s(k_1, \ldots, k_t)]$ para algum $s;\ (1 \leq s \leq u)$. Vamos mostrar que

$$bol_j(k_1, \ldots, k_t) \text{ implica } bol_q(k_1, \ldots, k_t)$$

que, por definição é igual a:

$$\forall \text{ valoração } V, \text{se } V[bol_j(k_1, \ldots, k_t)] = 1 \text{ então } V[bol_q(k_1, \ldots, k_t)] = 1.$$

Mas, se $V[bol_j(k_1, \ldots, k_t)] = 1$ temos que $V[bol_q^s(k_1, \ldots, k_t)] = 1$. Como $bol_q^s(k_1, \ldots, k_t)$ é uma sub-expressão conjuntiva completa da FND completa de $bol_q(k_1, \ldots, k_t)$, temos que

$$\forall \text{ valoração } V, \text{se } V[bol_q^s(k_1, \ldots, k_t)] = 1 \text{ então} V[bol_q(k_1, \ldots, k_t)] = 1.$$

Como $V[bol_j(k_1, \ldots, k_t)] = V[bol_q^s(k_1, \ldots, k_t)]$, portanto temos que

$$\forall \text{ valoração } V, \text{se } V[bol_j(k_1, \ldots, k_t)] = 1 \text{ então} V[bol_q(k_1, \ldots, k_t)] = 1$$

ou seja:

$$bol_j(k_1, \ldots, k_t) \text{ implica } bol_q(k_1, \ldots, k_t).$$

Logo é válido que

$$bol_j(k_1, \ldots, k_t) = bol_q^s(k_1, \ldots, k_t) \text{ para algum } s;\ (1 \leq s \leq u)$$
$$\Downarrow$$
$$bol_j(k_1, \ldots, k_t) \text{ implica } bol_q(k_1, \ldots, k_t).$$

Concluímos então que os modelos Ψ^b e Ψ^{vb} são equivalentes.

Capítulo 5

Uma Proposta de Representação de alguns Modelos de RI Modernos

Neste capítulo descrevemos os seguintes modelos de RI: modelo de redes de crença para combinar múltiplas fontes de evidências, modelo de RI baseado em ontologia para *Web* semântica e o modelo *sTerm*. Em seguida, é proposta uma representação destes modelos na estrutura funcional. Além disso, como consequência das propostas apresentadas, construímos novos modelos e realizamos comparações algébricas entre os modelos.

5.1 Representação $RC \rightarrow \Psi^{rc}$, $V_{rc} \rightarrow \Psi^{vrc}$, $\Psi^{rc} \simeq \Psi^{vrc}$

Nesta seção, iniciamos apresentando a definição formal de redes bayesianas e introduzimos o modelo de redes de crença para RI. Em seguida descrevemos o modelo de redes de crença para combinar múltiplas fontes de evidências. Finalmente, representamos este modelo na estrutura funcional usando a notação λ-cálculo e comparamos com o modelo vetorial estendido.

5.1.1 Redes Bayesianas

As redes bayesianas consistem em um grafo acíclico direcionado de dependências, cujos nós representam variáveis randômicas proposicionais ou constantes, e as arestas indicam as relações de dependência entre os nós [40]. No grafo, uma aresta significa que o primeiro tem influência direta sobre

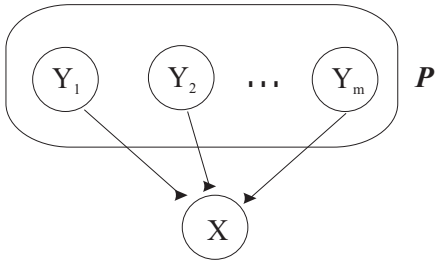


Fig. 5.1: Nós pais de um nó em uma rede bayesiana

o segundo. Esta influência é quantificada através de uma função de distribuição de probabilidade condicional correlacionando os estados de cada nó com os estados dos nós pais. A idéia principal é que, para descrever um modelo do mundo real, não é necessário usar uma enorme tabela de probabilidade conjunta na qual são listadas as probabilidades de todas as combinações possíveis de eventos.

A topologia da rede pode ser vista como uma base de conhecimento abstrata, representando a estrutura dos processos causais no domínio. Uma vez que a topologia da rede está definida, é necessário especificar as probabilidades condicionais para os nós que participam diretamente das relações de dependência. Cada nó possui uma tabela de probabilidade condicional que quantifica a influência que os nós pais têm sobre cada nó filho.

O princípio fundamental é que as dependências conhecidas entre as variáveis aleatórias do domínio são declaradas explicitamente na rede e que a distribuição conjunta de probabilidade pode ser inferida a partir dessas dependências. Os relacionamentos entre os nós, indicados pelos arcos direcionados do grafo, representam dependências causais ou as influências diretas entre as variáveis do domínio. A intensidade dessas influências ou dependências é expressa por probabilidades condicionais associadas aos arcos do grafo. As dependências declaradas são utilizadas para inferir as crenças (probabilidades) associadas a todas as variáveis da rede.

Sejam X e Y duas variáveis randômicas e x e y seus respectivos valores. Usamos X e Y para referenciar as variáveis randômicas e os nós na rede associados às variáveis. Um arco direcionado de Y, o nó pai, para X, o nó filho, representa a influência da variável Y sobre a variável X, que é quantificada pela probabilidade condicional $P(x|y)$.

Seja ***P*** o conjunto de nós pais de um nó X, como mostra a Figura 5.1. Seja **p** um conjunto de valores para todas as variáveis em ***P*** e seja x um valor da variável X. A influência de ***P*** sobre X pode ser modelada por qualquer função F tal que $\sum_x F(x, \mathbf{p}) = 1$ e $0 \leq F(x, \mathbf{p}) \leq 1$. A função $F(x, \mathbf{p})$

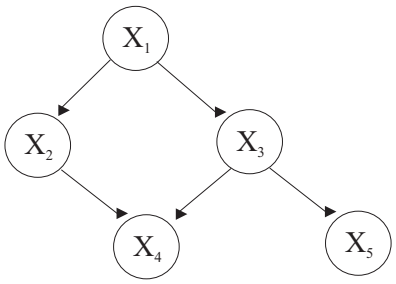


Fig. 5.2: Exemplo de uma rede bayesiana

fornece uma quantificação numérica para $P(x|\mathbf{p})$

Uma rede bayesiana fornece uma completa descrição sobre o seu domínio. Cada entrada na distribuição de probabilidade conjunta pode ser calculada da informação na rede, denotamos por $P(x_1, \ldots, x_r)$, onde r é o número total de variáveis. O valor da entrada é dado pela fórmula $P(x_1, \ldots, x_r) = \prod_{i=1}^{r} P(x_i|Pais(X_i))$.

Um exemplo de uma rede bayesiana de distribuição de probabilidade conjunta $P(x_1, x_2, x_3, x_4, x_5)$ é mostrado na Figura 5.2. O nó X_1, nó raiz, é um nó sem pais cuja distribuição de probabilidade é a probabilidade *a priori* $P(x_1)$. Dado o valor da variável X_1, as variáveis X_2 e X_3 são independentes. Dados os valores das variáveis X_2 e X_3, as variáveis X_4 e X_5 são independentes. Devido à independência declarada na Figura 5.2, a distribuição de probabilidade conjunta pode ser calculada como $P(x_1, x_2, x_3, x_4, x_5) = P(x_1) \cdot P(x_2|x_1) \cdot P(x_3|x_1) \cdot P(x_4|x_2, x_3) \cdot P(x_5|x_3)$

Na rede bayesiana, os nós recebem parâmetros numéricos. Estes parâmetros contêm graus de crença de acordo com algum conhecimento.

Uma vantagem das redes bayesianas é o poder de síntese de representação dos relacionamentos probabilísticos. É necessário considerar somente o conhecimento de independência entre as variáveis em um domínio. As independências declaradas no tempo de modelagem são usadas para inferir crenças para todas as variáveis na rede. O mecanismo de inferência é exponencial em alguns casos, mas é eficiente em muitas situações práticas, particularmente para o contexto de RI. Outra grande vantagem das redes bayesianas é que elas podem ser naturalmente estendidas por evidências geradas a partir de fontes independentes de conhecimento.

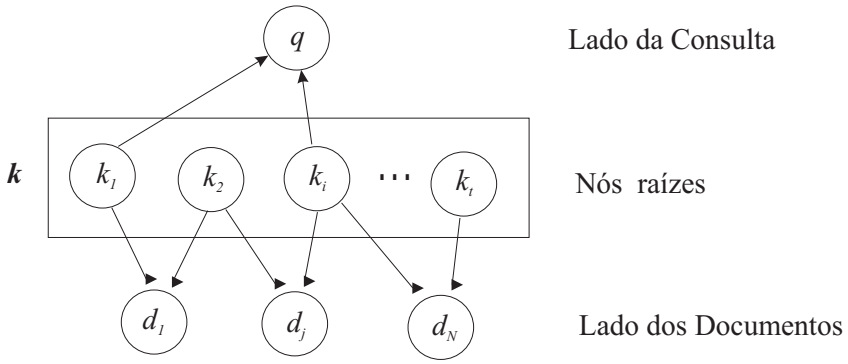


Fig. 5.3: Rede bayesiana para uma consulta q composta pelos termos k_1 e k_i

5.1.2 O Modelo de Redes de Crença para RI

Nesta seção, descrevemos o modelo de redes de crença proposto em [42] que fornece uma visão epistemológica do problema de RI e interpreta probabilidades como graus de crença.

Redes bayesianas permitem combinar características de diferentes modelos em um mesmo esquema representacional, e por isso, conseguem modelar os eventos e a interdependência de três componentes básicos em RI: palavras-chaves (ou termos), documentos e consultas. Em um modelo probabilístico, cada um desses componentes pode ser visto como um evento. Esses eventos não são independentes, visto que, por exemplo, a ocorrência de um termo influenciará na ocorrência de um documento. Usando uma rede bayesiana, podemos modelar esses eventos e suas interdependências.

Em um SRI tradicional, documentos são indexados por termos. O conjunto de todos os termos é interpretado como um universo **K**. Seja t o número de termos em uma coleção, então $\mathbf{K} = \{k_1, k_2, \ldots, k_t\}$.

Cada termo k_i está associado à uma variável randômica, denotada por K_i. Esta variável é 1 para indicar que um evento associado com o termo k_i ocorreu. Para simplificar a notação, escrevemos $P(k_i)$ ao invés de $P(K_i = 1)$ e $P(\bar{k_i})$ ao invés de $P(K_i = 0)$.

Um documento d_j é modelado como um conjunto composto de termos selecionados que ocorrem em seu texto. Uma variável randômica D_j é associada com cada documento d_j, e uma variável Q com a consulta q do usuário. Considerando também que as consultas são compostas de termos. A rede resultante desta modelagem é mostrada na Figura 5.3.

Nesta rede, cada nó d_j modela um documento, o nó q modela a consulta do usuário, e o nó k_i modela os termos na coleção. A instanciação dos nós raízes separa os nós documentos do nó da

consulta, tornando-os mutuamente independente. Então, na rede de crença da Figura 5.3, dizemos que há o lado da consulta e o lado dos documentos.

A similaridade entre um documento d_j e a consulta q pode ser interpretada como a probabilidade do documento d_j ser observado dado que a consulta q foi observada. Então, usando a lei de $Bayes$ e a regra da probabilidade total, calculamos a similaridade $P(d_j|q)$ como:

$$P(d_j|q) = \eta \sum_{\forall \mathbf{k}} P(d_j|\mathbf{k}) \times P(q|\mathbf{k}) \times P(\mathbf{k}) \tag{5.1}$$

onde $\eta = 1/P(q)$, como usada em [40], é uma constante de normalização. Esta equação é a expressão genérica para o *ranking* de um documento d_j em relação a consulta q, em um modelo de rede de crença.

Para representar qualquer um dos modelos tradicionais de RI, usando a rede da Figura 5.3, precisamos apenas definir as probabilidades $P(d_j|\mathbf{k})$,$P(q|\mathbf{k})$ e $P(\mathbf{k})$ apropriada.

Em seguida, descrevemos o modelo de redes de crença para combinar múltiplas fontes de evidências. A apresentação deste modelo segue [39]. A partir do modelo funcional obtido, encontramos o modelo vetorial equivalente. O resultado é a representação de redes bayesianas para combinar múltiplas evidências em um modelo vetorial equivalente.

5.1.3 Modelo de Redes de Crença para Combinar Múltiplas Fontes de Evidências

O modelo de redes bayesianas pode ser usado para combinar múltiplas fontes de evidências, tais como evidências baseadas em palavras-chaves associadas com o conteúdo dos documentos, o texto do conteúdo de *links* e a informação da análise de *links* entre documentos da coleção. Isto pode ser obtido através da adição de novas arestas, nós e probabilidades à rede bayesiana original apresentada na Figura 5.3. O modelo extendido pode ser observado na Figura 5.4.

O modelo que será descrito é um modelo genérico de redes bayesianas para combinar múltiplas fontes de evidências. Este modelo é uma extensão do modelo de redes de crença proposto em [1] e é aqui denotado por redes-crença (rc). A diferença é que o operador usado na composição de

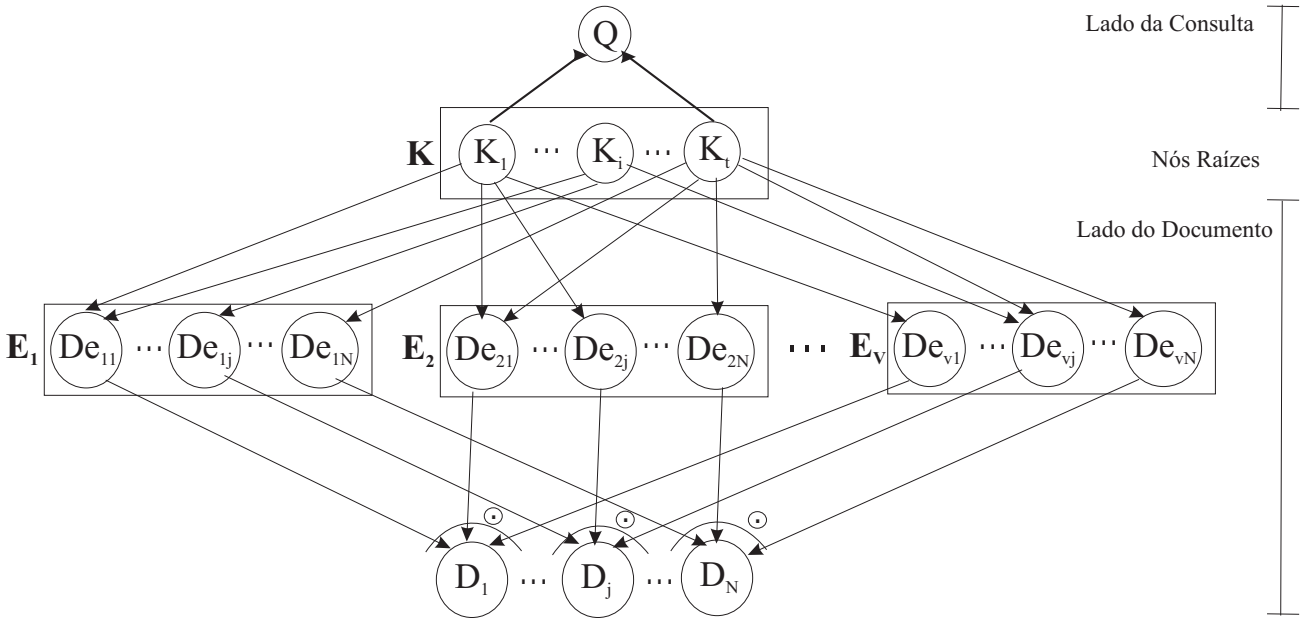


Fig. 5.4: Modelo de rede de crença para combinar múltiplas fontes de evidências

documentos é generalizado. Em [1] o operador é de disjunção, aqui ele é genérico e representado por $\odot$. A Figura 5.4 ilustra esta rede bayesiana generalizada para combinar múltiplas evidências.

Na rede bayesiana da Figura 5.4, o nó Q modela a consulta do usuário e o conjunto de nós **K** modela o conjunto de palavras-chave na coleção de documentos. Os conjuntos de nós $\mathbf{E}_1, \ldots, \mathbf{E}_v$ representam v evidências modeladas na rede. Os arcos ligando os nós de **K** aos documentos em $\mathbf{E}_1, \ldots, \mathbf{E}_v$ indicam que os termos da consulta induzem crença aos nós de documentos de acordo com a evidência representada, que pode ser por exemplo *links*. Para representar uma nova fonte de evidência E_i nesta rede, novos nós $De_{i,j}$ são associados com cada documento D_j no conjunto resposta para a consulta Q. O conjunto de nós **K** é usado para modelar a ocorrência de termos na consulta Q que induzem valores de crença em cada um dos nós dos conjuntos $\mathbf{E}_1, \ldots, \mathbf{E}_v$. O nó D_j representa a combinação de todas as evidências modeladas.

O *ranking* de um documento é calculado como a probabilidade $P(d_j|q)$, como a seguir:

$$P(d_j|q) = \eta \sum_{\forall \mathbf{k}} P(d_j|\mathbf{k}) \times P(q|\mathbf{k}) \times P(\mathbf{k}) \tag{5.2}$$

onde η é uma constante de normalização. Detalhes da derivação dessa expressão podem ser encontrados em [48]. A probabilidade condicional $P(d_j|\mathbf{k})$ depende de múltiplas evidências combinadas

através do operador $\odot$, que pode ser os operadores disjuntivo, conjuntivo e noisy-OR.

Para o operador disjuntivo, temos a equação:

$$P(d_j|\mathbf{k}) = 1 - (1 - P(de_{1j}|\mathbf{k})) \times (1 - P(de_{2j}|\mathbf{k})) \times \cdots \times (1 - P(de_{vj}|\mathbf{k})) \tag{5.3}$$

onde $P(de_{ij}|\mathbf{k})$ é o valor calculado para cada evidência E_i em relação ao documento d_j que denotamos aqui como E_{ij}. E_{ij} pode ser, por exemplo, o peso da parte de conteúdo do documento d_j, calculado pelo modelo clássico vetorial, ou o grau de *hub* ou o grau de *autoridade* do documento d_j [33]. E $P(q|\mathbf{k})$ é definido por:

$$P(q|\mathbf{k}) = \begin{cases} 1, & \text{se } \mathbf{q} = \mathbf{k} \\ 0, & \text{caso contrário} \end{cases} \tag{5.4}$$

Substituindo cada $P(de_{ij}|\mathbf{k})$ por E_{ij} na Eq.(5.3), e substituindo as Eq.(5.3) e (5.4) na Eq.(5.2), definindo a probabilidade *a priori* $P(\mathbf{k})$ como constante e considerando que a constante η não influencia no resultado final do *ranking*, podemos definir a *função similaridade* como:

$$sim(d_j, q) = 1 - (1 - E_{1j})(1 - E_{2j}) \ldots (1 - E_{vj}) \tag{5.5}$$

Observe que qualquer evidência E_i pode ser ignorada, atribuindo E_{ij} = 0. Note que esta função de similaridade não satisfaz a propriedade de simetria, pois $sim(d_j, q) \neq sim(q, d_j)$.

Analogamente, para o operador conjuntivo, tem-se a multiplicação dos valores de cada evidência como mostrado na seguinte função:

$$sim(d_j, q) = E_{1j} \times E_{2j} \cdots \times E_{vj} \tag{5.6}$$

Note que se para qualquer evidência e_i, E_{ij} = 0, então $sim(d_j, q)$ = 0, ignorando todos as outras evidências. Por isto o operador conjuntivo não é muito utilizado na prática.

A combinação no modelo usando os operadores disjuntivo e conjuntivo não considera a hipótese *a priori* sobre a importância de cada fonte de evidência. As probabilidades a serem combinadas dependem somente das características dos algoritmos e dos parâmetros usados. Entretanto, o modelo

pode ser modificado para permitir a inserção de pesos. Isto pode ser realizado utilizando o operador noisy-OR (maiores detalhes sobre este operador podem ser encontrados em [40]). Então, tem-se a seguinte equação para a função de similaridade:

$$sim(d_j, q) = 1 - (1 - W_1 \times E_{1j})(1 - W_2 \times E_{2j}) \ldots (1 - W_v \times E_{vj}) \tag{5.7}$$

onde $W_1 \ldots W_v$ são os pesos atribuídos para cada evidência $E_1, \ldots, E_v$, respectivamente. Estes pesos podem ser definidos pelo usuário, podem depender ou não da consulta ou podem ser automaticamente calculados.

Para simplificar a notação, seja $R_{j,q}$ a função de *ranking* do modelo vetorial de D_j com relação à consulta Q. A informação fornecida pelo modelo vetorial pode ser incluída como uma evidência fazendo E_{1j} = $R_{j,q}$.

5.1.4 Modelo λ-redes-crença (Ψ^{rc})

A representação do modelo de redes de crença na estrutura funcional é mostrado aqui. Para representar o modelo de redes de crença genérico que combina múltiplas fontes de evidências usando o operador disjuntivo na estrutura funcional baseada em λ-cálculo, definimos o modelo funcional $\Psi^{rc} = \langle [df_1^{rc}, ..., df_n^{rc}], [qf^{rc}], \Delta^{rc} \rangle$. O modelo bayesiano com múltiplas evidências pode ser representado na estrutura funcional por:

- $df_j^{rc} = [peso_{e1_j}, peso_{e2_j}, \ldots, peso_{ev_j}]$, onde $peso_{e1_j}(k_i)$ é a função peso, análoga a do modelo vetorial, de cada termo do documento d_j e $peso_{e2_j} \ldots peso_{ev_j}$ são funções que definem valores para as evidências $e_2, \ldots, e_v$ associadas com o documento d_j,respectivamente.

- $qf^{rc} = [peso_{e1_q}, peso_{e2_q}, \ldots, peso_{ev_q}]$, onde $peso_{e1_q}(k_i)$ é a função peso, análoga a do modelo vetorial, de cada termo da consulta q e os outros termos funcionais são definidos de forma análoga aos documentos funcionais.

- A função similaridade Δ^{rc} é dada por

$$\lambda x \lambda y.sim^{rc}(x, y) \text{ onde}$$

$$sim^{rc} = \lambda\vec{f}\lambda\vec{g}.((1-(1-R_{j,q})\cdot(1-f_2\cdot g_2)\ldots(1-f_v\cdot g_v)))$$

tal que,

$$\lambda\vec{f} \equiv \lambda f_2\lambda f_3\ldots\lambda f_v,$$

$$\lambda\vec{g} \equiv \lambda g_2\lambda g_3\ldots\lambda g_v$$

e $R_{j,q}$ é dado pela Equação

$$\frac{\sum_{i=1}^{t} peso_{e1_j}(k_i).peso_{e1_q}(k_i)}{\sqrt{\sum_{i=1}^{t} peso_{e1_j}(k_i)^2}\times\sqrt{\sum_{i=1}^{t} peso_{e1_q}(k_i)^2}}.$$

Sendo assim, temos

$\Delta^{rc}(df_j^{rc}, qf^{rc}) = (\lambda x\lambda y).sim^{rc}(x,y))(df_j^{rc}, qf^{rc})$
$= sim^{rc}(x,y)\{x \leftarrow df_j^{rc}, y \leftarrow qf^{rc}\}$
$= sim^{rc}(df_j^{rc}, qf^{rc})$
$= sim^{rc}([peso_{e2_j},\ldots,peso_{ev_j}],[peso_{e2_q},\ldots,peso_{ev_q}])$
$= sim^{rc}(peso_{e2_j},\ldots,peso_{ev_j},peso_{e2_q},\ldots,peso_{ev_q})$
$= \lambda\vec{f}\lambda\vec{g}.(1-(1-R_{j,q})\cdot(1-f_2\cdot g_2)\ldots(1-f_v\cdot g_v)))$
$(peso_{e2_j},\ldots,peso_{ev_j},peso_{e2_q},\ldots,peso_{ev_q})$
$= (1-(1-R_{j,q})\cdot(1-f_2\cdot g_2)\ldots(1-f_v\cdot g_v)))$
$\{f_2 \leftarrow peso_{e2_j}, g_2 \leftarrow peso_{e2_q},\ldots,f_v \leftarrow peso_{e2_v}, g_v \leftarrow peso_{e2_v})\}$
$= (1-(1-R_{j,q})\cdot(1-peso_{e2_j}\cdot peso_{e2_q})\ldots(1-peso_{ev_j}\cdot peso_{ev_q}))$

Portanto,

$$\Delta^{rc}(df_j^{rc}, qf^{rc}) = 1-(1-R_{j,q})(1-peso_{e2_j}peso_{e2_q})\ldots(1-peso_{ev_j}peso_{ev_q}).$$

Nesta função de similaridade (Δ^{rc}), a propriedade de normalização é satisfeita, pois consideramos que os valores das evidências são normalizados ($0 \leq peso_{e1j},\ldots,peso_{evj} \leq 1$) e a função $R_{j,q}$ também. A propriedade de reflexividade também é válida, pois se $R_{j,q} = 1$, então $\Delta^{rc}(df_j^{rc}, df_j^{rc}) = 1$ e a propriedade de simetria também é válida, pois $\Delta^{rc}(df_j^{rc}, qf^{rc}) = \Delta^{rc}(qf^{rc}, df_j^{rc})$. Portanto, a função similaridade Δ^{rc} é do tipo Δ_{nrs}.

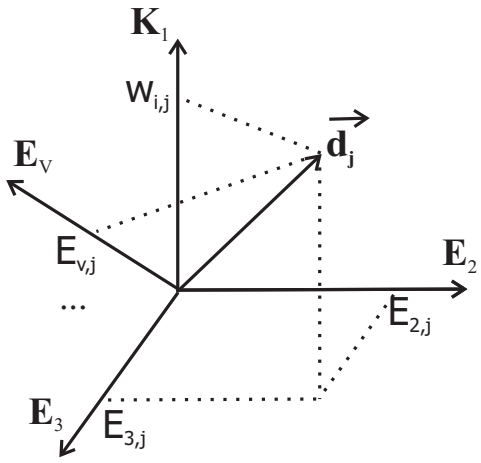


Fig. 5.5: Modelo genérico vetorial para combinação de múltiplas fontes de evidências

5.1.5 Modelo vetorial-redes-crença (V_{rc})

Apresentamos aqui um modelo vetorial estendido para combinar múltiplas evidências equivalente à rede bayesiana anterior proposto em [39], em seguida representamos este modelo na estrutura funcional baseado em λ-cálculo. No modelo clássico vetorial, o conjunto de termos $\{k_i | 1 \leq i \leq t\}$ formam os eixos do modelo vetorial. Os documentos e consultas são representados como vetores no espaço: $\vec{d_j} = (w_{1j}, w_{2j}, \ldots, w_{tj})$ e $\vec{q} = (w_{1q}, w_{2q}, \ldots, w_{tq})$, respectivamente.

Apresentamos um modelo que combina informações de múltiplas fontes de evidências através de uma extensão do modelo de espaço vetorial. Para isso, o espaço vetorial é estendido adicionando $v - 1$ novos eixos, onde $v - 1$ é o número de novas evidências. A Figura 5.5 mostra este modelo vetorial para combinação de múltiplas fontes de evidências.

Neste caso, a equação da função de similaridade é:

$$sim(d_j, q) = 1 - (1 - R_{j,q})(1 - E_{2q} \times E_{2j}) \ldots (1 - E_{vq} \times E_{vj})$$

onde $R_{j,q}$ é calculado pelo cosseno do modelo vetorial, $E_{2q}, \ldots, E_{vq}$ são os valores de cada evidência $e_2, \ldots, e_v$ associado à consulta q e $E_{2j}, \ldots, E_{vj}$ são os valores de cada evidência $e_2, \ldots, e_v$ associados ao documento d_j, respectivamente.

A seguir representamos este modelo na estrutura funcional.

5.1.6 Modelo λ-vetorial-redes-crença (Ψ^{vrc})

Representamos o modelo vetorial para combinar múltiplas fontes de evidências apresentado anteriormente na estrutura funcional baseado no λ-cálculo. Para representar o modelo genérico vetorial para combinação de múltiplas evidências na estrutura funcional, definimos o modelo funcional $\Psi^{vrc} = \langle [df_1^{vrc}, ..., df_n^{vrc}], [qf^{vrc}], \Delta^{vrc} \rangle$, onde:

- $df_j^{vrc} = \{peso_{e1_j}, peso_{e2_j}, \ldots, peso_{ev_j}\}$, onde $peso_{e1_j}$ é a função peso calculada pelo modelo vetorial de cada termo do documento d_j e $peso_{e2_j} \ldots peso_{ev_j}$ são funções que definem valores para as evidências $e_2, \ldots, e_v$ associadas com o documento d_j,respectivamente.

- $qf^{vrc} = \{peso_{e1_q}, peso_{e2_q}, \ldots, peso_{ev_q}\}$, onde $peso_{e1_q}$ e os outros termos são definidos de forma análoga aos documentos funcionais.

- Função similaridade é dada por

$$\Delta^{vrc}(df_j^{vg}, qf^{vrc}) = 1 - (1 - R_{j,q})(1 - peso_{e2_j} peso_{e2_q}) \ldots (1 - peso_{ev_j} peso_{ev_q}) \qquad (5.8)$$

Verificamos que os modelos funcionais Ψ^{rc} e Ψ^{vrc} são equivalentes por construção. Existe a função bijetora identidade ϕ e as duas propriedades de equivalência são satisfeitas, pois possuem a mesma função de similaridade. Os modelos Ψ^{rc} e Ψ^{vrc} são equivalentes, e geram o mesmo *ranking*.

5.2 Representação $OWS \rightarrow \Psi^{ows}$

5.2.1 Modelo de RI baseado em Ontologia para Web Semântica (OWS)

Recentemente, pesquisas sobre ontologia têm-se expandido para outras áreas do conhecimento como forma de integração de sistemas de informação aplicáveis em vários campos, tais como: engenharia, comércio eletrônico, educação, tecnologia, recuperação da informação entre outros. O motivo pelo qual a ontologia vêm ganhando popularidade é a promessa de que um domínio do conhecimento possa ser representado computacionalmente, viabilizando a comunicação entre pessoas e computadores, automaticamente, de forma inteligente. Dentro deste contexto está inserida uma das metas

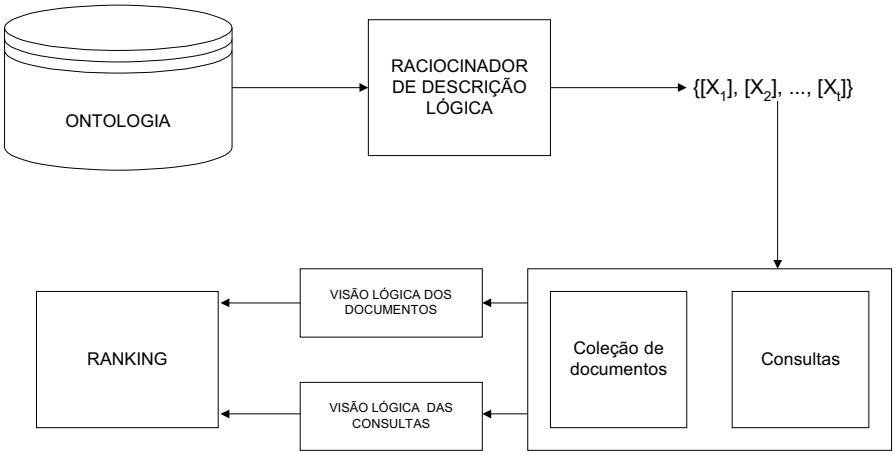


Fig. 5.6: Principais partes do modelo de RI baseado em Ontologia para a *Web* semântica.

da *web* semântica, que visa estruturar máquinas de buscas para melhorar o resultado das pesquisas trazendo compreensão, isto é, significado exato das palavras e as relações conceituais entre elas [5]. A *Web* semântica permite melhorar a organização através da estruturação dos dados com a utilização de linguagens que usam semântica. A *web* semântica também está baseada no uso de ontologias que fornecem os vocabulários necessários para permitir a definição das relações entre os conceitos dos sistemas. O método básico para construção da *Web* semântica é utilizar os termos definidos na ontologia como metadados para marcação do conteúdo da *Web*, dando significado as suas respectivas páginas. Desta forma, [30] acreditam que os modelos de recuperação de informação baseados em ontologias para *Web* semântica possam atender as necessidades de informação do usuário com maior precisão.

Nesta seção descrevemos um modelo de RI baseado em Ontologia para a *Web* Semântica (OWS). A apresentação deste modelo segue [30]. Vale destacar que não estamos interessados nos resultados apresentados pelo modelo, mas apenas na modelagem deste. Sendo assim, serão descritas as principais partes que compõem o modelo WSO (para maiores detalhes veja [30]). A Figura 5.6 mostra este modelo.

A geração da ontologia

O método de construção de ontologias proposto em [30], é baseado na possibilidade de reuso de conceitos definidos em outras ontologias. O processo de construção de novas ontologias é, portanto,

condicionado à análise e à adaptação de conceitos de outras ontologias (ontologias de domínio). Uma ontologia de domínio descreve o vocabulário relativo a um domínio específico através da especialização de conceitos presentes na ontologia (veja detalhes em [7]). No modelo OWS, o processo de construção da ontologia segue as etapas descritas a seguir:

1ª **etapa (captura)-** Definir os conceitos e relacionamentos (domínio da ontologia). Nesta etapa é realizado um levantamento dos termos que serão relevantes para descrever os conceitos e relacionamentos da ontologia.

2ª **etapa (formalização/tradução)-** Nessa etapa se formaliza o modelo conceitual da fase anterior através de uma linguagem formal para descrição de ontologias.

3ª **etapa (integração) -** Nessa etapa é realizada a integração da ontologia em desenvolvimento com outras ontologias (ontologias de domínio).

A ontologia do modelo OWS é construída a partir destas etapas. Na Figura 5.7 destaca-se as etapas de tradução e integração. A idéia por trás destas etapas é a seguinte: como diferentes domínios de ontologias na *Web* semântica são usualmente codificados em diferentes linguagens de ontologia, é necessário um mecanismo de tradução. Este mecanismo utiliza, por exemplo, a linguagem *OWL Lite* como a linguagem de ontologia padrão para formalização dos conceitos. Sendo assim, as ontologias de domínio codificadas em outras linguagens são traduzidas em ontologias de domínio codificadas em *OWL Lite*. Vale destacar, que durante a tradução, é inevitável que se perca algum conhecimento, ou seja, nem todas as sentenças codificadas em outras linguagens de ontologia podem ser traduzidas em sentenças codificadas em *OWL Lite*. Após esta tradução, é feita uma integração destas ontologias de domínio codificadas em OWL Lite para obter uma ontologia de domínio sobre todas estas ontologias de domínio. Esta ontologia de domínio é considerada uma ontologia. Quando uma nova ontologia de domínio é adicionada, esta deve primeiramente ser traduzida em um domínio codificado em *OWL Lite*, e então ser integrado com a ontologia existente. Desta forma tem-se uma nova ontologia que caracteriza o conceito de uma forma mais precisa e compreensiva.

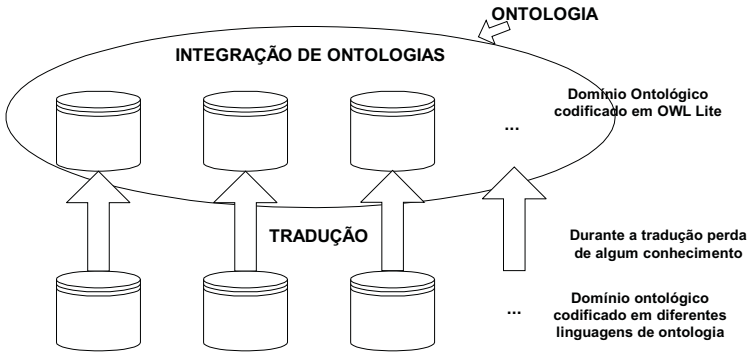


Fig. 5.7: Tradução e integração de ontologias

A geração das classes de equivalências dos termos semânticos

Como descrito anteriormente, o modelo OWS utiliza a linguagem de ontologia *OWL Lite* para construção da ontologia. Esta linguagem oferece primitivas de modelagem que permitem a construção de hierarquias (relacionamentos entre conceitos), classes (conceitos) e propriedades. Sendo assim, depois que a ontologia é construída, os termos (por exemplo, classes, propriedades) definidos na ontologia são usados como metadados para marcação do conteúdo da *Web*. Estes marcações semânticas são denominadas termos semânticos para a recuperação de informação baseada em ontologia. Os termos semânticos são identificados através de URLs.

Os raciocinadores de descrição lógica (por exemplo, FaCT e RACER [2, 28]) são utilizados para solucionar problemas de inferência lógica em *OWL Lite*, ou seja, permitem uma maior expressividade do conhecimento através do uso dos conectivos lógicos [30]. A partir destes raciocinadores lógicos e dos termos semânticos definidos na ontologia, as classes de equivalência são obtidas através de inferências na ontologia (mais detalhes em [2, 28]).

Antes de apresentar o modelo de RI baseado em ontologia, considere as definições a seguir:

Definição 5.1 (relação de equivalência)). *Uma relação binária no conjunto **X** é um subconjunto **R** de $\mathbf{X} \times \mathbf{X}$. Usualmente, escrevemos $a \sim b$ para denotar que $(a, b) \in \mathbf{R}$. O símbolo $\sim$ representa uma relação em **R**. Um relação binária $\sim$ no conjunto **X** é chamada de uma relação de equivalência se para todo $x, y, z \in \mathbf{X}$, temos:*

- $x \sim x$ *(propriedade reflexiva)*

- $x \sim y$ *implica* $y \sim x$ *(propriedade simétrica)*

- $x \sim y$ *e* $y \sim z$ *implica* $x \sim z$ *(propriedade transitiva)*

Definição 5.2 (classes de equivalência). *Se* $\sim$ *é uma relação de equivalência em* $\boldsymbol{X}$*, então para* $x \in \boldsymbol{X}$ *podemos definir o conjunto* $[x] := \{y \in \boldsymbol{X} : x \sim y\}$*. O conjunto* $[x]$ *é uma classe de equivalência.*
Segue da definição de uma relação de equivalência que para todo $x, y \in \mathbf{X}$, temos:

- $x \in [x]$,

- $[x] \cap [y] = \emptyset$ e $[x] \neq [y]$

Para qualquer $x \in \mathbf{X}$, o conjunto $[x]$ é chamado de classe de equivalência contendo x, e x é chamado de representativo de $[x]$.

Tendo em vista as definições, temos portanto que: a relação de equivalência $\sim$ no conjunto X é um subconjunto R de $X \times X$, tal que: $\forall X_i \in X$, temos que $X_i \sim X_j \Leftrightarrow (X_i, X_j) \in R$. Além disso, a relação de equivalência $\sim$ deve satisfazer as três propriedades. Como $\sim$ é uma relação de equivalência em X, então para $X_i \in X$ podemos definir o conjunto $[X_i] := \{X_j \in X : X_i \sim X_j\}$. O conjunto $[X_i]$ é uma classe de equivalência. O conjunto de todas as classes de equivalências é dado por: $\mathbf{C} = \{[X_1], [X_2], \ldots, [X_t]\}$ $(1 \leq i \leq t)$.

Modelo de RI baseado em ontologia para Web Semântica

Considere inicialmente o conjunto dos termos semânticos de uma coleção denotado por $\mathbf{X} = \{X_1, ..., X_u\}$ $(1 \leq i \leq u)$. A construção de uma ontologia define uma relação de equivalência R no conjunto $\mathbf{X}$. Esta relação de equivalência é tal que: $X_i \ R \ X_j$ se e somente se têm o mesmo significado semântico na ontologia. Classes de equivalência, segundo R, são denotadas por $[X_i]$. Logo, $[X_i] = \{X_j \mid X_i \ R \ X_j\}$. Neste contexto, dado que X_j, $X_s \in [X_i]$ então X_j é denotado por X_{ij} e X_s por X_{is}.

Seja d_j um documento, e x, y e z três termos diferentes que estão presentes em d_j. Se a marcação semântica para x, y e z é X_{i1}, X_{i2} e X_{i3}, respectivamente, então embora x, y e z tenham aparências diferentes, do ponto de vista semântico elas são iguais, pois podem ser representadas pelo mesmo termo semântico X_i.

Considere uma coleção de documentos na *Web* semântica dada por **D** $= \{d_1, d_2, \ldots, d_n\}$, tal que $d_j \in$ **D** $(1 \leq j \leq n)$. A freqüência total de todos os elementos de uma classe de equivalência $[X_i](1 \leq i \leq t)$ que aparecem em um documento d_j é dada por f_{ij} (ou seja, o número de vezes que todos os elementos da classe de equivalência $[X_i]$ aparecem em todas as marcações semânticas do documento d_j. A tabela 5.1 mostra esta freqüência total.

Classes de equivalências	d_1	d_2	...	d_j	...	d_n
$[X_1]$	f_{11}	f_{12}	$\ldots$	f_{1j}	$\ldots$	f_{1n}
$[X_2]$	f_{21}	f_{22}	$\ldots$	f_{2j}	$\ldots$	f_{2n}
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$[X_i]$	f_{i1}	f_{i2}	$\ldots$	f_{ij}	$\ldots$	f_{in}
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$[X_t]$	f_{t1}	f_{t2}	$\ldots$	f_{tj}	$\ldots$	f_{tn}

Tab. 5.1: A frequência total dos elementos de uma classe de equivalência

Como no esquema tf-idf do modelo vetorial, tem-se as seguintes fórmulas:

$$\begin{cases} tf_{ij} & = \frac{f_{ij}}{max\{f_{1j}, f_{2j}, \ldots, f_{tj}\}} \\ idf_i & = log_2 \frac{n}{n_i} \\ w_{ij} & = tf_{ij} \times idf_i \end{cases}$$

Onde n é o número de documentos e n_i é o número de documentos nos quais os elementos da classe de equivalência $[X_i]$ aparecem.

Utilizando estas fórmulas, calcula-se todos os pesos, desta forma a visão lógica de todos os documentos em **D** pode ser dada por:

$$d_j = (w_{1j}, w_{2j}, \ldots, w_{tj})$$

Esta visão lógica reflete a semântica dos termos semânticos, de tal modo que estes podem representar satisfatoriamente os documentos.

Para uma consulta q, obtém-se a visão lógica de q de acordo com o conjunto das classes de equivalências dos termos semânticos : $\{[X_1], [X_2], \ldots, [X_t]\}$. Assim, uma consulta é dada por:

$$q = (w_{1q}, w_{2q}, \ldots, w_{tq})$$

Esta visão lógica reflete a semântica dos termos semânticos, de tal modo que estes podem representar satisfatoriamente a consulta.

A similaridade entre q e d_j pode ser calculada de acordo com a seguinte função de *ranking*:

$$sim(d_j, q) = \frac{1}{\sum_{i=1}^{t}(w_{iq} - w_{ij})^2}.$$

Esta função de ranking define uma ordenação entre os documentos da coleção considerando a consulta q.

O modelo OWS tem como principal vantagem lidar com a semântica dos termos, em vez de sua sintaxe. Por outro lado, a praticabilidade deste modelo ainda é questionada. Como a ontologia é muito grande, de que maneira pode-se efetivamente encontrar os termos semânticos adequados na ontologia para marcação do conteúdo da *Web* [30]?

5.2.2 Modelo λ-ontologia (Ψ^{ows})

Nesta seção propomos uma representação funcional do modelo de RI baseado em Ontologia para *Web* Semântica (OWS), descrito anteriormente. Antes de representarmos este modelo na estrutura funcional, serão definidos alguns conceitos importantes para a representação e estabelecido algumas premissas.

Para simplificar, partimos do pressuposto que a ontologia já tenha sido criada, ou seja, os termos (conceitos, classes, hierarquias) que representam a ontologia estão definidos. Estes termos, denominados termos semânticos, são as marcações semânticas para os termos presentes nos documentos. Seja X_s um termo semântico, o conjunto formado pelos termos semânticos é dado por: $\mathbf{X} = \{X_1, X_2, \ldots, X_u\}$ $(1 \leq s \leq u)$.

A partir do conjunto **X** e uma relação de equivalência sobre **X**, o raciocinador lógico descritivo(RLD) irá gerar as classes de equivalências.

Notação: utilizaremos a notação $relEqui$ na estrutura funcional para denotar a relação de equivalência utilizada pelo RLD para gerar as classes de equivalências.

Exemplo: Seja $\mathbf{X} = \{X_1, ..., X_u\}$ o conjunto dos termos semânticos e $2^{\mathbf{X}}$ o conjunto de sub-conjuntos

de **X** chamado conjunto potência. A função $syn : \mathbf{X} \rightarrow 2^{\mathbf{X}}$, por exemplo, é a função sinônima tal que, dado um termo semântico, retorna o conjunto de sinônimos de cada termo semântico. Dado X_i, se $syn(X_i) = \{X_{i1}, ..., X_{is}\}$, então o conjunto de sinônimos do termo X_i é igual a $\{X_{i1}, \ldots, X_{is}\}$.
A partir de $syn(X_i)$ obtemos a relação de equivalência syn e suas classes de equivalência $[X_i]$, tal que $X_i\ syn\ X_j \Leftrightarrow syn(X_j) \in [X_i]$. Observe que a relação syn satisfaz as três propriedades:

- $X_i\ syn\ X_i$ - (reflexiva)
- $X_i\ syn\ X_j$ implica $X_j\ syn\ X_i$ - (simétrica)
- $X_i\ syn\ X_k\ e\ X_k\ syn\ X_j$ implica $X_i\ syn\ X_j$ - (transitiva)

Portanto, a função sinônima syn define uma relação de equivalência em X.

Depois de definidas as classes de equivalências a partir de uma função, utilizamos os termos semânticos (definidos na ontologia) para marcação dos termos dos documentos. Seja a coleção de documentos $D = \{d_1, d_2, ..., d_n\}$, tal que $d_j \in D$ para $1 \leq j \leq n$. O conjunto de todos os termos sintáticos presentes em D é dado por: $\mathbf{K} = \{k_1, \ldots, k_x\}\ (1 \leq i \leq x)$. Definimos então a função $indexSem$ onde dado o conjunto de termos **K** e o conjunto de termos semânticos **X**, a função especifica qual termo semântico serve como marcação para um determinado k_i.

Suponha que todas as classes de equivalências dos termos semânticos para a coleção **D** seja representada pelo conjunto $\mathbf{C} = \{[X_1], [X_2], \ldots, [X_t]\}\ (1 \leq i \leq t)$. De acordo com as equações (tf_{ij}, idf_i, w_{ij}) descritas anteriormente no modelo OWS, temos uma nova visão lógica dos documentos e consultas.

Podemos agora representar o modelo OWS na estrutura funcional. Para representar um modelo na estrutura funcional, é necessário definir um modelo funcional Ψ que o represente. A representação do modelo OWS na estrutura funcional é denotada por Ψ^{ows}, onde $\Psi^{ows} = \langle [df_1^{ows}; ...; df_n^{ows}], [qf^{ows}], \Delta^{ows}] \rangle$ e

- $df_j^{ows} = [pesoSem_j, relEqui, indexSem]$. Os documentos funcionais são listas contendo os termos funcionais $pesoSem_j, relEqui$ e $indexSem$. Cada um destes termos serão definidos em seguida:

– O termo funcional $relEqui$ é uma relação de equivalência em **X**, a partir da qual as classes de equivalências $[X_1], ..., [X_t]$ são geradas. $relEqui$ é uma função dada por: $\lambda x \lambda y.relEqui(x, y))$ tal que:

$$relEqui(X_i, X_j) = true \Leftrightarrow (X_i, X_j) \in \mathbf{R} \text{ onde } \mathbf{R} \text{ é um subconjunto de } \mathbf{X}.$$

Um exemplo de relação de equivalência é a função sinônima.

– O termo funcional $indexSem$ especifica qual termo semântico serve como marcação para um determinado termo sintático k_i.

– O termo funcional $pesoSem_j$ aplicado ao termo semântico X_i define o peso do termo semântico no documento d_j. A função $pesoSem_j$ é denotada por: $\lambda x.pesoSem_j\ x$.

$$(\lambda x.pesoSem_j\ x)\ (X_i) = pesoSem_j X_i$$

• $qf^{ows} = [pesoSem_q]$. As consultas funcionais são listas contendo apenas o termo funcional $pesoSem_q$.

Dado um termo semântico X_i definido na ontologia, o termo funcional $pesoSem_q$ para este termo define o peso do termo X_i na consulta q, denotado por: $\lambda x.pesoSem_q\ x$.

$$(\lambda x.pesoSem_q\ x)\ (X_i) = pesoSem_q X_i$$

• A função similaridade Δ^{ows} é dada por

$$\lambda x \lambda y.sim^{ows}(x, y)$$

onde

$$sim^{ows} = \lambda f \lambda g. \frac{1}{\sqrt{\sum_{i=1}^{t}(f(X_i) - g(X_i))^2}}$$

Sendo assim, temos

$$\begin{aligned}
\Delta^{ows}(qf^{ows}, df_j^{ows}) &= (\lambda x \lambda y.sim^{ows}(x,y))(qf^{ows}, df_j^{ows}) \\
&= sim^{ows}(x,y)\{x \leftarrow qf^{ows}, y \leftarrow df_j^{ows}\} \\
&= sim^{ows}(qf^{ows}, df_j^{ows}) \\
&= sim^{ows}(pesoSem_q, pesoSem_j) \\
&= \lambda f \lambda g. \frac{1}{\sqrt{\sum_{i=1}^{t}(f(X_i)-g(X_i))^2}}(pesoSem_q, pesoSem_j) \\
&= \frac{1}{\sqrt{\sum_{i=1}^{t}(f(X_i)-g(X_i))^2}}\{f \leftarrow pesoSem_q, g \leftarrow pesoSem_j\} \\
&= \frac{1}{\sqrt{\sum_{i=1}^{t}(pesoSem_q(X_i)-pesoSem_j(X_i))^2}}
\end{aligned}$$

Portanto,

$$\Delta^{ows}(qf^{ows}, df_j^{ows}) = \frac{1}{\sqrt{\sum_{i=1}^{t}(pesoSem_q(X_i) - pesoSem_j(X_i))^2}}$$

Note que a função similaridade $\Delta^{ows}(qf^{ows}, df_j^{ows})$ é apenas simétrica, pois temos que $\Delta^{ows}(qf^{ows}, df_j^{ows}) = \Delta^{ows}(df_j^{ows}, qf^{ows})$. As propriedades normalização e reflexividade não são satisfeitas, pois a função similaridade não é normalizada e $pesoSem_j$ deve ser diferente de $pesoSem_q$. Portanto, a função similaridade é do tipo Δ_s.

5.3 Representação $ST \rightarrow \Psi^{st}$, $V_{st} \rightarrow \Psi^{vst}$

5.3.1 Modelo *sTerm* (ST)

A recuperação de informação em documentos XML vem despertando um grande interesse dos pesquisadores. Este interesse é justificado porque a XML vem se tornando um padrão, amplamente aceito, para o armazenamento, troca e apresentação de dados [11]. Vários modelos têm sido propostos na literatura para trabalhar nessa área, como por exemplo [10, 16, 47]. Nesta seção, descrevemos o modelo *sTerm* que é um modelo para recuperação de informação em documentos XML. A apresentação deste modelo segue [47]. Primeiro, descrevemos a idéia geral e em seguida, apresentamos os detalhes deste modelo.

A base do modelo *sTerm* é o modelo vetorial clássico. Os conceitos tais como documento, consulta e termos são estendidos para uma interpretação estruturada. Uma consulta no modelo vetorial clássico é uma lista de palavras chaves. No modelo *sTerm*, é adicionada estrutura nas palavras chaves de tal forma que as consultas possam ser interpretadas como árvores rotuladas. Documentos XML são interpretados como árvores rotuladas, também. Uma coleção de documentos é modelada como uma única árvore, e cada subárvore como um documento lógico. A raiz da árvore da consulta determina a noção de documentos admissíveis: todo documento lógico, cujo nó raiz é igual ao nó raiz da consulta, é um candidato potencial a ser retornado como resultado. Este documento é comparado com a consulta, e atribui-se um grau de similaridade que determina a sua posição no *ranking*. O grau de similaridade é calculado utilizando a distribuição dos termos estruturados (*s-terms*). Termos estruturados são, essencialmente, subárvores da consulta e dos documentos. O número de ocorrências de um termo estrutural dentro de um documento lógico e o número de documentos lógicos que contém o termo estrutural são contados, normalizados e utilizados para computar o peso de um termo, de forma análoga ao que ocorre no modelo vetorial. Os pesos são utilizados para construir os vetores documentos. Por outro lado, os pesos do vetor consulta podem ser definidos pelo usuário. Os vetores consulta e documento são comparados utilizando critérios próprios do modelo.

ÁRVORES E SUAS PROPRIEDADES

Definição 5.3 (árvore). *Uma árvore é uma estrutura* $T = (\boldsymbol{N}, \boldsymbol{A}, raiz(T))$ *que consiste de um conjunto finito de nós* ***N****, um conjunto finito de arestas* ***A****, e um nó* $raiz(T) \in \boldsymbol{N}$ *que forma o nó raiz de* T.

O conjunto de arestas é uma relação binária em N onde cada par $(u, v) \in \mathbf{A}$ estabelece uma relação entre dois nós de **N**. Se $(u, v) \in \mathbf{A}$, então u é *pai* de v, e v é *filho* de u. O conjunto **A** deve satisfazer as seguintes condições:

1. a raiz não tem pai, e
2. todo nó da árvore exceto o nó raiz tem exatamente um pai.

O conjunto de filhos de um nó $u \in \mathbf{N}$ é denotado por:

$$filhos(u) = \{v \in \mathbf{N} \mid (u, v) \in \mathbf{A}\}.$$

Um nó sem filho é chamado *nó folha*. Nós que não são folhas são chamados *nós internos*.

Definição 5.4 (caminho). *Um caminho em uma árvore é uma sequência de nós $u_1, u_2, \ldots, u_n$ tal que, para todo par u_i, u_{i+1} de nós consecutivos existe uma aresta $(u_i, u_{i+1}) \in \boldsymbol{A}$.*

Um nó u é chamado *ancestral* de v (e v *descendente* de u) se e somente se existe um caminho $u = u_1, \ldots, u_n = v$, onde $n > 1$.

Definição 5.5 (subárvore). *Seja a árvore $T = (\boldsymbol{N}, \boldsymbol{A}, raiz(T))$ e o nó $u \in N$. Uma subárvore $T' = (\boldsymbol{N'}, \boldsymbol{A'}, u)$ de T é uma árvore, onde*

$\boldsymbol{N'} = \{u\} \cup \{v \mid v \in \boldsymbol{N} \wedge v$ é descendente de u em $T\}$ e

$\boldsymbol{A'} = \boldsymbol{A} \cap (\boldsymbol{N'} \times \boldsymbol{N'})$.

Denotamos $T' \lhd T$ se e somente se T' é uma subárvore de T.

Definição 5.6 (árvore rotulada). *Uma árvore rotulada é uma árvore $T = (\boldsymbol{N}, \boldsymbol{A}, raiz(T))$ junto com uma função rótulo: $\boldsymbol{N} \rightarrow \Sigma$, onde Σ é um conjunto finito de rótulos.*

Definição 5.7 (árvores isomórficas). *Duas árvores $T = (\boldsymbol{N}, \boldsymbol{A}, raiz(T))$ e $T' = (\boldsymbol{N'}, \boldsymbol{A'}, raiz(T'))$ são isomórficas se e somente se, existe um mapeamento bijetivo f entre $\boldsymbol{N}$ e $\boldsymbol{N'}$ tal que f preserva os rótulos, a relação pai-filho e a ordem dos filhos de T.*

DOCUMENTOS XML E CONSULTAS COMO ÁRVORES ROTULADAS

Utilizando a noção descrita anteriormente sobre árvores, cada documento XML físico é mapeado para uma árvore rotulada. Os elementos são representados por um nó que tem o nome do elemento como rótulo. Sequências de textos são decompostas em palavras. Cada palavra é mapeada para um nó folha rotulado com a respectiva palavra. Atributos são mapeados para dois nós que são pai e filho um do outro: o nome do atributo é o rótulo do nó pai, e o valor do atributo é o rótulo do filho. A Figura 5.8 mostra o mapeamento de um exemplo de documento XML para uma árvore com seus respectivos rótulos.

Definição 5.8 (coleção de documentos XML). *Uma coleção C de documentos XML é uma árvore rotulada com um nó raiz único.*

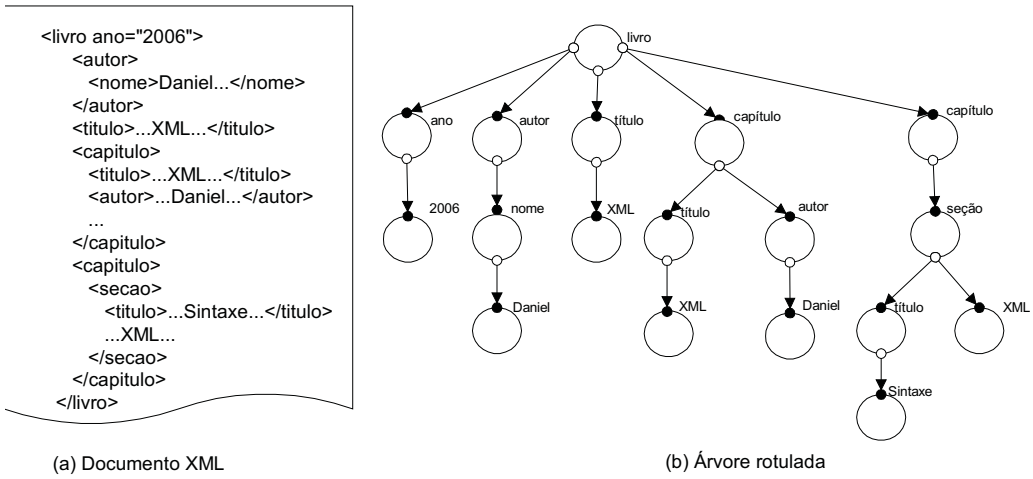


Fig. 5.8: Mapeamento de um documento XML para um árvore rotulada

Exemplo: Na Figura 5.9 (parte (b)) tem uma coleção de documentos com o nó raiz [biblioteca].

Modelos de RI tradicionais lidam com o conceito de documento estático que são limitados por arquivos físicos. Entretanto, no modelo *sTerm*, esta noção física de um documento é trocada por uma mais flexível, a noção lógica.

Definição 5.9 (documento lógico). *Um documento lógico D em uma coleção C é uma subárvore de C.*

Exemplo: Na parte da coleção descrita na Figura 5.9 não somente a subárvore enraizada pelo nó [livro] (correspondendo a representação de um documento XML físico) é um documento lógico, mas também todas as suas subárvores, ou seja, a subárvore com o nó raiz [secao] ou a subárvore consistindo do nó rotulado ["2006"].

Definição 5.10 (consulta XML). *Uma consulta Q é uma árvore rotulada.*

Definição 5.11 (tipo). *Seja $T = (\boldsymbol{N}, \boldsymbol{A}, raiz(T))$ uma consulta ou um documento. O tipo de T é o rótulo de sua raiz: $tipo(T) = rotulo(raiz(T))$.*

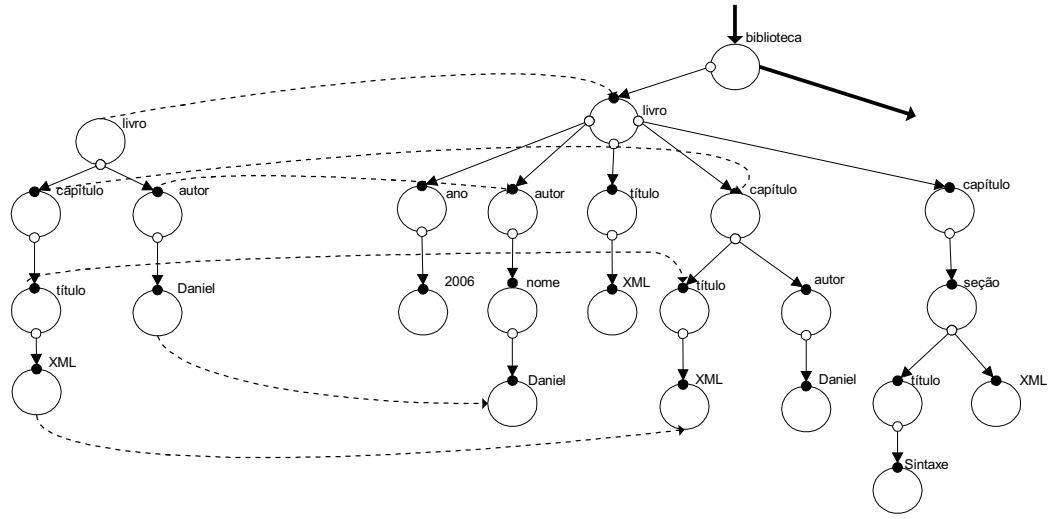
Em [47], assume-se que o usuário ao definir o tipo da consulta defina também o tipo de documentos lógicos a serem recuperados.

Definição 5.12 (documentos admissíveis). *O conjunto de documentos admissíveis para a consulta Q em relação a coleção de documentos C é dado por*

$$D^{tipo(Q)} = \{D \mid D \triangleleft C \ \wedge \ tipo(D) = tipo(Q)\}.$$

TREE MATCHING

Anteriormente, mostramos como interpretar os dados e a consulta por meio de árvores. Com esta interpretação, o problema de responder a uma consulta pode ser mapeado como um problema de "*correlação*" de uma árvore de consulta em uma árvore de dados.



(a) Árvore de consulta (b) Parte de uma coleção de documentos

Fig. 5.9: Correlação entre uma árvore de consulta e uma árvore de dados

Definição 5.13 (correlação). *Seja a consulta $Q = (\boldsymbol{M}, \boldsymbol{A}, raiz(Q))$ e $C = (\boldsymbol{N}, \boldsymbol{E}, raiz(C))$ a coleção de documentos. Uma função f de $\boldsymbol{M}$ para $\boldsymbol{N}$ é uma correlação de Q em C se e somente se para todo $u, v \in \boldsymbol{M}$ é válido que*

1. $rotulo(u) = rotulo(f(u))$,

2. u *é ancestral de* $v \Rightarrow f(u)$ *é ancestral de* $f(v)$.

Definição 5.14 (casamento). *Seja* Q *uma consulta,* C *uma coleção de documentos e* f *uma correlação de* Q *em* C*. Então* $f(raiz(Q))$ *é um casamento de* Q *em* C*.*

O documento lógico, com o nó raiz do casamento de Q em C, é uma resposta para a consulta. Exemplo: A Figura 5.9 mostra uma correlação da árvore de consulta (à esquerda) e da árvore de dados (à direita). Existe somente um documento admissível descrito para a consulta: a subárvore com o nó raiz [livro]. As linhas pontilhadas mostram uma correlação f que define o casamento entre a consulta e o documento.

Definição 5.15 (termo estrutural). *Toda árvore rotulada sobre um dado conjunto de rótulos* Σ *é um termo estrutural.*

Na definição seguinte, a noção de casamentos introduzidos para consultas e documentos, é estendida para termos e documentos.

Definição 5.16 (ocorrências de termos). *Seja* T *um termo estrutural.*

- *Se existe um termo estrutural* T' *isomórfico a* T *tal que* $T' \lhd Q$ *para uma consulta* Q*, então* T *ocorre em* Q*. A raiz de* T' *é denominada de uma ocorrência de* T *em* Q*.*

- *Se* T *casa com um documento* D*, então* T *ocorre em* D*. O casamento de* T *em* D *é uma ocorrência.*

Notação: uma árvore como a da Figura 5.9 (parte(a)) é denotada pela expressão

$$livro[capitulo[titulo["XML"]], autor["Daniel"]].$$

Exemplo: Seis termos estruturais ocorrem na consulta da Figura 5.9: as duas subárvores consistindo somente dos nós folhas ["XML"] e ["Daniel"], respectivamente. Em seguida, as três subárvores enraizadas pelos nós internos [titulo], [capitulo] e [autor], respectivamente. Finalmente, a própria árvore de consulta é um termo estrutural que ocorre na consulta. O documento XML tem mais ocorrências de termos estruturais tais como: autor[nome], ["Daniel"],livro["Daniel",titulo[XML]].

CALCULANDO OS PESOS

Definição 5.17 (frequência do termo). *Seja T um termo estrutural, e D um documento lógico. Seja $freq_T(D)$ o número de ocorrências de T em D, e $maxfreq(D)$ o número máximo de ocorrências de um termo qualquer em D. A frequência de um termo $tf_{T,D}$ de T em D é definida por*

$$tf_{T,D} = \frac{freq_T(D)}{maxfreq(D)}.$$

Exemplo: A Figura 5.9 (parte (b)) mostra um documento lógico do tipo [livro] como parte de uma coleção. O número máximo de ocorrências de um termo neste documento é três, pois temos três nós ["XML"] e três nós [titulo]. O termo estrutural autor["Daniel"] tem portanto uma frequência de $\frac{2}{3}$ no documento lógico [livro].

A frequência inversa de documentos idf_T^t é limitada pelos documentos de um certo tipo, visto que uma consulta somente seleciona documentos que casam com seu tipo.

Definição 5.18 (frequência inversa de documentos). *Seja T um termo estrutural, e t um tipo. Seja $|D^t|$ o número de documentos do tipo t, e n_T o número de documentos em D^t casados por T. A frequência inversa idf_T^t de T é definida por*

$$idf_T^t = log\frac{|D^t|}{n_T} + 1$$

Exemplo: A coleção descrita na Figura 5.9 mostra dois documentos lógicos do tipo $capitulo$, portanto $|D^{capitulo}|$ é igual a 2 nesta parte da coleção. O termo estrutural $T = autor[" Daniel"]$ ocorre em um destes documentos lógicos. Isto significa que $n_T = 1$, e

$$idf_T^{capitulo} = log\ 2\ \ + 1\ =\ 1.30.$$

Definição 5.19 (peso do documento). *Seja $D \in D^t$ um documento de tipo t. O peso $w_{T,D}^t$ do termo estrutural T em D é definido como*

$$w_{T,D}^t = tf_{T,D}.idf_T^t.$$

SIMILARIDADE

Definição 5.20 (vetor documento). *O vetor peso v_D para o documento D do tipo t é definido por*

$$v_D = (w^t_{T_1,D}, \ldots, w^t_{T_m,D})$$

Definição 5.21 (vetor consulta). *O vetor peso v_Q de uma consulta Q é definida por*

$$v_Q = (w_{T_1,Q}, \ldots, w_{T_m,Q}),$$

onde $w_{T_i,Q} > 0$ se e somente se T_i ocorre em Q, e $w_{T_i,Q} = 0$ caso contrário.

Definição 5.22 (similaridade). *Seja Q uma consulta com o vetor consulta v_Q, e D um documento com o vetor documento v_D. A similaridade $sim(Q, D)$ entre v_Q e v_D é definida por*

$$sim(Q, D) = \begin{cases} v_Q \cdot v_D, & \text{se } D \in D^{type(Q)} \\ 0, & \text{caso contrário} \end{cases}$$

onde $\cdot$ denota o produto interno entre os vetores.

O modelo *sTerm* apresenta algumas vantagens: (i) fornece ao usuário condições de contextualizar a sua necessidade de informação; (ii) permite casamentos parciais, não sendo necessário que o usuário tenha um conhecimento completo da estrutura dos documentos. A principal desvantagem do modelo *sTerm* é que a indexação dos termos é feita em tempo de consulta, consequentemente, apresenta um aumento no tempo de processamento da consulta [47].

5.3.2 Modelo λ-*sTerm* (Ψ^{st})

Propomos a seguir uma representação do modelo *sTerm* na estrutura funcional. Para representar um modelo na estrutura funcional, é necessário definir um modelo funcional Ψ que o represente. Seja $\mathbf{T} = T_1, T_2, ..., T_m$ o conjunto de todos os termos estruturais não isomórficos que tem ocorrências na coleção de documentos. A representação do modelo *sTerm* na estrutura funcional é denotada por Ψ^{st}, onde $\Psi^{st} = \langle [df^{st}_1; ...; df^{st}_n], [qf^{st}], \Delta^{st}] \rangle$ e

- $df^{st}_j = [peso^{st}_j, tipo]$, onde $peso^{st}_j : \mathbf{T} \rightarrow \mathbb{R}$ é uma função unária que retorna o peso de um termo estrutural no documento lógico D_j. A λ-expressão $peso^{st}_j$ é dada por: $\lambda x.peso^{st}_j(x)$. Quando aplicada ao termo estrutural T_i, retorna o peso do termo estrutural T_i no documento lógico D_j e denotamos por $peso^{st}_j(T_i)$. Além disso, temos que cada documento lógico D_j, tem um tipo que é dado pelo rótulo de sua raiz. A função tipo é definida por $tipo : \mathbf{D} \rightarrow \Sigma$, tal que $\mathbf{D} =$

$\{D_1, \ldots, D_n\}$ e Σ é um conjunto finito de rótulos. Sendo assim, $tipo(D_j) = rotulo(raiz(D_j))$. Portanto, temos que os documentos funcionais são listas contendo os termos funcionais $peso_j^{st}$ e $tipo$.

- $qf^{st} = [peso_q^{st}, tipo_q]$, onde $peso_q^{st} : \mathbf{T} \rightarrow \mathbb{R}$ é uma função unária que retorna o peso de um termo estrutural na consulta q. A λ-expressão $peso_q^{st}$ é dada por: $\lambda x.peso_q^{st}(x)$. Quando aplicada ao termo estrutural T_i, retorna o peso do termo estrutural T_i na consulta q e denotamos por $peso_q^{st}(T_i)$, onde $peso_q^{st} \geq 0$ se e somente se T_i ocorre em q, e $peso_q^{st} = 0$ caso contrário. Além disso, temos que cada consulta q, tem um tipo que é dado pelo rótulo de sua raiz. A função tipo é definida por $tipo_q : \mathbf{Q} \rightarrow \Sigma$, tal que $\mathbf{Q}$ é um conjunto de consultas e Σ é um conjunto finito de rótulos. Sendo assim, $tipo_q(q) = rotulo(raiz(q)$. Portanto, temos que as consultas funcionais são listas contendo os termos funcionais $peso_q^{st}$ e $tipo_q$.

- A função similaridade Δ^{st} é dada por

$$\lambda x \lambda y.sim^{st}(x, y) \text{ onde}$$

$$sim^{st} = \lambda f \lambda g \lambda h \lambda h_1. \left((g = h_1) \rightarrow \sum_{i=1}^{m} f(T_i).h(T_i) \mid 0 \right)$$

Sendo assim, temos

$$\begin{aligned}
\Delta^{st}(df_j^{st}, qf^{st}) &= (\lambda x \lambda y.sim^{st}(x, y))(df_j^{st}, qf^{st}) \\
&= sim^{st}(x, y)\{x \leftarrow df_j^{st}, y \leftarrow qf^{st}\} \\
&= sim^{st}(df_j^{st}, qf^{st}) \\
&= sim^{st}([peso_j^{st}, tipo], [peso_q^{st}, tipo_q]) \\
&= sim^{st}(peso_j^{st}, tipo, peso_q^{st}, tipo_q) \\
&= \lambda f \lambda g \lambda h \lambda h_1.((g = h_1) \rightarrow \textstyle\sum_{i=1}^{m} f(T_i).h(T_i) \mid 0)\ (peso_j^{st}, tipo, peso_q^{st}, tipo_q) \\
&= \lambda f \lambda g \lambda h \lambda h_1.((g = h_1) \rightarrow \textstyle\sum_{i=1}^{m} f(T_i).h(T_i) \mid 0) \\
&\quad \{f \leftarrow peso_j^{st}, g \leftarrow tipo, h \leftarrow peso_q^{st}, h_1 \leftarrow tipo_q\} \\
&= ((tipo = tipo_q) \rightarrow \textstyle\sum_{i=1}^{m} peso_j^{st}(T_i).peso_q^{st}(T_i) \mid 0)
\end{aligned}$$

Portanto,

$$\Delta^{st}(df_j^{st}, qf) = \begin{cases} \sum_{i=1}^{m} peso_j^{st}(T_i).peso_q^{st}(T_i), & \text{se } tipo = tipo_q \\ 0, & \text{caso contrário} \end{cases}$$

Note que podemos normalizar a função similaridade dividindo cada vetor consulta e documento pela sua norma, de tal forma que $0 \leq \Delta^{st}(df_j^{st}, qf^{st}) \leq 1$. Para tal, a função sim^{st} é modificada para

$$\lambda f \lambda g \lambda h \lambda h_1. \left((g = h_1) \rightarrow \frac{\sum_{i=1}^{m} f(T_i).h(T_i)}{\sqrt{\sum_{i=1}^{m} f(T_i)^2} \times \sqrt{\sum_{i=1}^{m} h(T_i)^2}} \mid 0 \right).$$

Fazendo esta modificação temos que a propriedade de normalização e reflexividade são satisfeitas. Por outro lado, a propriedade da simetria não é satisfeita, pois a ocorrência de um termo estrutural na consulta é diferente da ocorrência de um termo estrutural no documento (veja definição de ocorrências de termos na seção 5.3.1), consequentemente $\Delta^{st}(df_j^{st}, qf^{st}) \neq \Delta^{st}(qf^{st}, df_j^{st})$. Portanto, considerando a normalização dos vetores consulta e documento, a função de similaridade $\Delta^{st}(df_j^{st}, qf^{st})$ é do tipo Δ_{nr}.

5.3.3 Modelo vetorial-estrutural (V_{vst})

Definimos um modelo vetorial estendido para recuperação de informação em documentos XML. No modelo clássico vetorial, o conjunto de termos $\{k_i | 1 \leq i \leq t\}$ formam os eixos do modelo vetorial. Os documentos e consultas são representados como vetores no espaço: $\vec{d_j} = (w_{1j}, w_{2j}, \ldots, w_{tj})$ e $\vec{q} = (w_{1q}, w_{2q}, \ldots, w_{tq})$, respectivamente.

Propomos nesta seção um modelo que utiliza a estrutura dos documentos através de uma extensão do modelo de espaço vetorial. Para isso, extendemos o espaço vetorial adicionando novos eixos, onde estes eixos são formados pelos termos estruturais (definidos a seguir). Para descrevermos nosso modelo, utilizamos algumas definições descritas no modelo *sTerm*.

Extensão do Vocabulário da Recuperação de Informação Tradicional

Os termos que formam o vocabulário da recuperação de informação tradicional não possuem informação de estrutura. Por outro lado, a estrutura em árvores dos documentos XML possibilita a estruturação de um texto, como foi visto no modelo *sTerm*. As folhas das árvores de um documento XML

são os termos que não possuem estrutura. A contextualização desses termos (folhas) é fornecida pelos ramos das árvores das quais fazem parte. Portanto, as folhas das árvores XML representam os termos da recuperação de informação tradicional e esses termos, associados aos seus respectivos ramos, são os termos que contêm informação de estrutura. Considere, por exemplo, a subárvore com raiz [secao] mostrada na Figura 5.8, o termo [Sintaxe] apresenta apenas informação de conteúdo (folha da árvore). O termo secao[titulo[Sintaxe]] contêm informação de contexto (secao[titulo[]]) e de conteúdo (Sintaxe). A idéia principal para inserir a noção de contextualização ao modelo vetorial é aumentar o seu domínio com termos que possuam informação de estrutura. Assim, essa informação de contexto poderá ser utilizada pelo SRI no cálculo da similaridade entre documentos e consultas.

Para um melhor entendimento do modelo proposto, são apresentadas a seguir, algumas definições de termos. Estas definições consideram os aspectos dos termos que têm ou não estrutura e como podem ser derivados das subárvores dos documentos XML.

Definição 5.23 (Termo complexo). *Um termo complexo Tc é uma subárvore ou um caminho de uma subárvore, que possui informação de estrutura e conteúdo.*

Exemplo: Considere as seguintes partes da árvore da Figura 5.8, apresentadas na Figura 5.10.

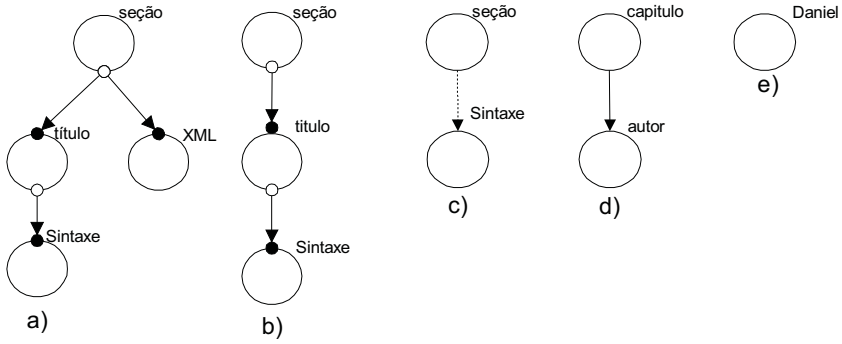


Fig. 5.10: Exemplos de termos complexos

- Em a) temos uma subárvore,portanto é um termo complexo.
- Em b) temos um caminho que possui informação de conteúdo e estrutura, logo é um termo complexo.

- A árvore representada em c) não é um caminho e nem uma subárvore, logo não é um termo complexo.
- O caminho em d) não é um termo complexo, pois só possui informação de estrutura.
- O nó "Sintaxe" em e) não é um termo complexo, pois só possui informação de conteúdo.

Definição 5.24 (Termo atômico). *Um termo atômico Ta é uma folha derivada dos termos complexos.*
Por exemplo, os termos [Sintaxe] e [XML] são termos atômicos derivados do termo complexo do item a) no exemplo acima.

Termo complexo e termo atômico são definidos com base no conceito de árvores e formam o domínio da RI semi-estrutural.

Definição 5.25 (Termo estrutural). *Um termo estrutural k^{st} é uma string produzida com base nos rótulos do termo complexo correspondente.*
Por exemplo, os termos complexos do exemplo acima (itens a) e b)) são associados aos termos estruturais: secao[titulo[Sintaxe],XML] e autor[nome[Daniel]], respectivamente.

Definição 5.26 (Termo plano). *Um termo plano k é um termo que não contêm informação de estrutura.*
É a definição de termo usualmente utilizada na RI tradicional.

A Figura 5.11 mostra a relação entre os termos atômicos, planos, complexos e estruturais. Observa-se que o domínio A é dividido em dois subconjuntos: o conjunto dos termos atômicos e o conjunto dos termos complexos. O domínio B também é dividido em dois subconjuntos: conjunto dos termos planos e conjunto dos termos estruturais. Além disso, existe uma relação biunívoca entre os elementos do domínio A e os elementos do domínio B.

O conjunto dos termos planos é o domínio da recuperação de informação tradicional, pois os termos planos não apresentam estrutura. Os conjuntos dos termos atômicos e termos complexos formam o domínio da recuperação de informação em dados semi-estruturados (domínio A). Para simular o modelo vetorial, [47] restringe, em tempo de consulta, o seu domínio de atuação para o conjunto dos termos atômicos. Os termos atômicos não possuem estrutura e têm um termo plano

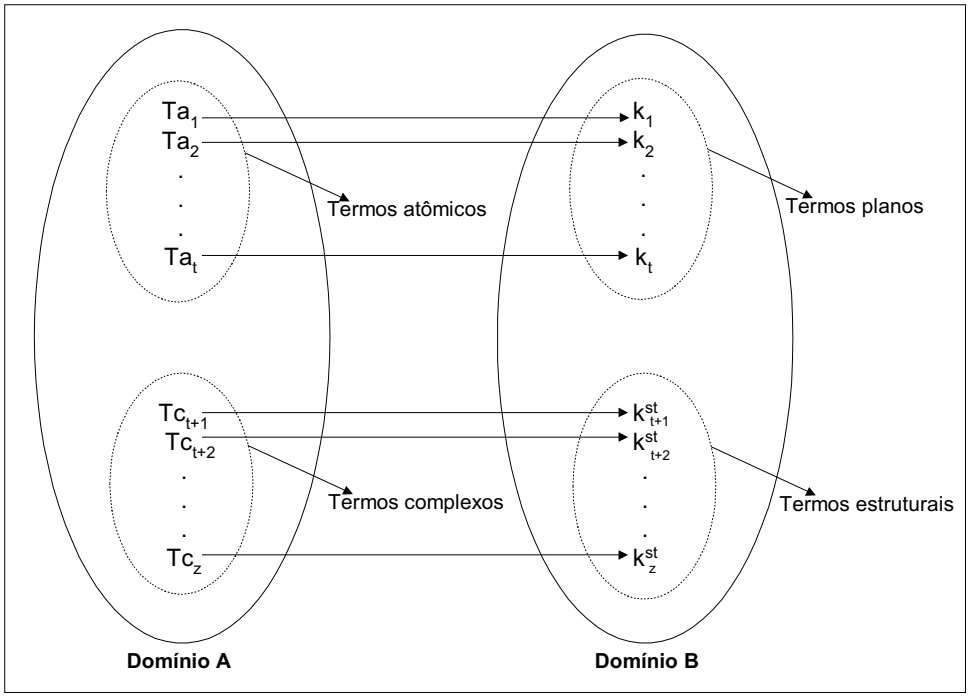


Fig. 5.11: Domínio dos termos

correspondente. Portanto, essa restrição possibilita que a abordagem descrita em [47] considere um domínio equivalente ao domínio da recuperação de informação tradicional.

A Figura 5.11 mostra, ainda, que os termos estruturais do domínio B são os correspondentes aos termos complexos do domínio A. O conjunto de termos complexos é composto por termos que possuem estrutura. Portanto, ao se estender o domínio dos modelos de recuperação de informação tradicional com termos que possuam informação de estrutura (termos estruturais), obtém-se o domínio B = $\{$termos planos$\} \cup \{$termos estruturais$\}$. Dessa maneira, é possível definir um modelo vetorial que utilize a informação de estrutura no cálculo do *ranking*.

Considere a base de termos $(k_1, \ldots, k_t)$ do modelo clássico vetorial. Sem perda de generalização, podemos ordenar a base de termos estruturais e planos $(k_1, \ldots, k_t, k^{st}_{t+1}, \ldots, k^{st}_z)$ tal que k_i é um termo plano para $1 \leq i \leq t$. O termo k^{st}_i para $i > t$ é um termo estrutural. Desta forma, podemos calcular os pesos dos termos estruturais e planos através da abordagem tf-idf do modelo vetorial. Portanto, podemos representar os documentos e as consultas através dos seguintes vetores: $\vec{d_j} = (w_{1,j}, \ldots, w_{t,j}, w_{t+1,j}, \ldots, w_{z,j})$ e $\vec{q} = (w_{1,q}, \ldots, w_{t,q}, w_{t+1,q}, \ldots, w_{z,q})$. A função similaridade pode

ser dada pelo produto interno entre os dois vetores. Sendo assim, apenas os termos estruturais e planos presentes na consulta tem pesos diferentes de 0, ou seja, são considerados para o cálculo da similaridade.

O modelo vetorial estendido apresenta algumas vantagens: (i) possibilidade de pré-indexação, o cálculo dos pesos dos termos estruturais não são realizados em tempo de consulta; (ii) alia a simplicidade e desempenho do modelo vetorial às vantagens da consulta contextualizada. A maior desvantagem do modelo vetorial estendido é que a quantidade de termos pode crescer exponencialmente. Apesar disso, hoje existe na literatura técnicas de indexação que podem ser utilizadas para contornar esta desvantagem, tais como [29, 53]. Vale destacar que a avaliação em relação ao desempenho e eficiência do modelo proposto não faz parte do escopo deste trabalho. Em seguida, representamos o modelo proposto na estrutura funcional e comparamos ele com o modelo *sTerm*.

5.3.4 Representação λ-vetorial-estrutural (Ψ^{vst})

Propomos a seguir uma representação do modelo vetorial-estrutural na estrutura funcional. Para representar um modelo na estrutura funcional, é necessário definir um modelo funcional Ψ que o represente. Seja $\mathbf{K} = \{k_1, \ldots, k_t, k_{t+1}^{st}, \ldots, k_z^{st}\}$ o conjunto de termos da coleção. A representação do modelo vetorial-estrutural na estrutura funcional é denotada por Ψ^{vst}, onde $\Psi^{vst} = \langle [df_1^{vst}; ...; df_n^{vst}], [qf^{vst}], \Delta^{vst}]) \rangle$ e

- $df_j^{vst} = [peso_j^{vst}]$. Os documentos funcionais são listas contendo apenas o termo funcional $peso_j^{vst}$. A função $peso_j^{vst}$, para um termo k_i ou k_i^{st}, define o peso do termo k_i ou k_i^{st} no documento d_j, denotada por $peso_j^{vst}(k_i)$ ou $peso_j^{vst}(k_i^{st})$.

- $qf^v = [peso_q^{vst}]$. As consultas funcionais são listas contendo apenas o termo funcional $peso_q^{vst}$. A função $peso_q^{vst}$, para um termo k_i ou k_i^{st}, define o peso do termo k_i ou k_i^{st} na consulta q, denotada por $peso_q^{vst}(k_i)$ ou $peso_q^{vst}(k_i^{st})$.

- A função similaridade Δ^{vst} é dada por

$$\lambda x \lambda y . sim^{vst}(x, y) \text{ onde}$$

$$sim^{v} = \lambda f \lambda g.\left(\sum_{i=1}^{t} f(k_i).g(k_i) \ + \ \sum_{i=t+1}^{z} f(k_i^{st}).g(k_i^{st})\right)$$

Sendo assim, temos

$$\begin{aligned}
\Delta^{vst}(df_j^{vst}, qf^{vst}) &= (\lambda x \lambda y.sim^{vst}(x,y))(df_j^{vst}, qf^{vst}) \\
&= sim^{vst}(x,y)\{x \leftarrow df_j^{vst}, y \leftarrow qf^{vst}\} \\
&= sim^{vst}(df_j^{vst}, qf^{vst}) \\
&= sim^{vst}([peso_j^{vst}], [peso_q^{vst}]) \\
&= sim^{vst}(peso_j^{vst}, peso_q^{vst}) \\
&= (\lambda f \lambda g.\sum_{i=1}^{t} f(k_i).g(k_i) \ + \ \sum_{i=t+1}^{z} f(k_i^{st}).g(k_i^{st}))\ (peso_j^{vst}, peso_q^{vst}) \\
&= (\sum_{i=1}^{t} f(k_i).g(k_i) \ + \ \sum_{i=t+1}^{z} f(k_i^{st}).g(k_i^{st}))\{f \leftarrow peso_j^{vst}, g \leftarrow peso_q^{vst}\} \\
&= \sum_{i=1}^{t} peso_j^{vst}(k_i).peso_q^{vst}(k_i) \ + \ \sum_{i=t+1}^{z} peso_j^{vst}(k_i^{st}).peso_j^{vst}(k_i^{st})
\end{aligned}$$

Portanto,

$$\Delta^{vst}(df_j^{vst}, qf^{vst}) = \sum_{i=1}^{t} peso_j^{vst}(k_i).peso_q^{vst}(k_i) \ + \ \sum_{i=t+1}^{z} peso_j^{vst}(k_i^{st}).peso_j^{vst}(k_i^{st})$$

Comparação Ψ^{st} e Ψ^{vst}

Tendo em vista a representação na estrutura funcional os modelos Ψ^{st} e Ψ^{vst} não são equivalentes. Uma prova desta não equivalência é dada no exemplo a seguir:

Exemplo: considere a Figura 5.12, onde apresentamos uma árvore de consulta (parte (a)) e a árvore de dados representando uma parte da coleção de documentos ((parte (b))).

Temos que no modelo funcional Ψ^{st}, o documento físico que tem como raiz o nó [livro] não é retornado no *ranking*, pois o tipo da consulta (rótulo da raiz da árvore [tese]) é diferente. Logo, a similaridade é 0. Por outro lado, no modelo funcional Ψ^{vst}, o documento físico que tem como raiz o nó livro é retornado no *ranking*, pois temos que a similaridade é maior que 0. Isto ocorre, porque no modelo funcional Ψ^{vst} o peso referente aos termos estruturais:

$$capitulo[titulo[XML]], titulo[XML], autor[Daniel], XML\ e\ Daniel$$

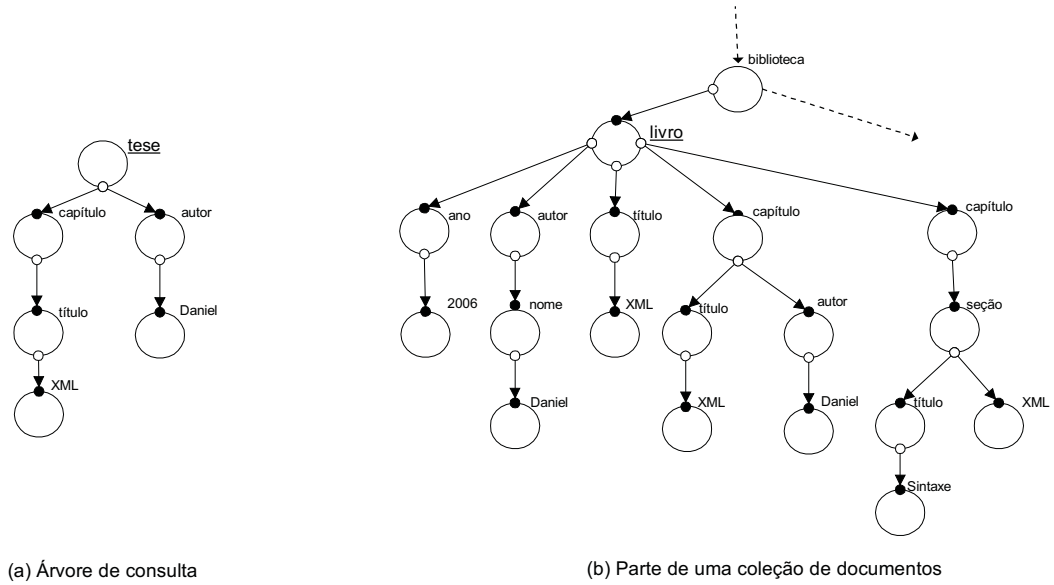


Fig. 5.12: Árvores de consulta e dados

no documento livro e na consulta tese são diferentes de zero. Assim tais termos são considerados no cálculo da similaridade, ou seja, o produto interno entre o vetor consulta (tese) e o vetor documento (livro) é maior que 0. Sendo assim, o *ranking* gerado pelos dois modelos são diferentes, ou seja, os modelos Ψ^{st} e Ψ^{vst} não são equivalentes.

No modelo Ψ^{vst} não consideramos o tipo da consulta e o tipo do documento na função ranking. Para construirmos um modelo vetorial (Ψ^{vstt}) equivalente a Ψ^{st} fazemos a seguinte modificação em Ψ^{vst}:

Acrescentamos o tipo em todos os termos da coleção. Por exemplo, considere a Figura 5.12. O termo estrutural (tese[capitulo[titulo["XML"]]]) gera os seguintes termos estruturais do tipo tese:

$$tese[capitulo[titulo[XML]]]_{tese}, capitulo[titulo[XML]]_{tese}, tese[titulo[XML]]_{tese},$$

$$tese[XML], titulo[XML]_{tese}, capitulo[XML]_{tese} \text{ e } XML_{tese}.$$

Ao fazermos este acréscimo do tipo, asseguramos que apenas os documentos, onde seus termos estruturais tem o mesmo tipo dos termos estruturais da consulta, são retornados no ranking. Sendo assim, no modelo vetorial Ψ^{vstt} a similaridade entre a consulta (do tipo [tese])e o documento físico (do tipo

livro), representados na Figura 5.12, é 0. Isto ocorre, porque os tipos dos termos da consulta e do documento são diferentes, ou seja, temos que todos os termos estruturais presentes na consulta são do tipo [tese], e estes termos não ocorrem no vetor documento. Por exemplo, temos que

$$capitulo[titulo[XML]]_{tese} \neq capitulo[titulo[XML]]_{livro}.$$

A representação do modelo vetorial Ψ^{vstt} é análoga a do modelo vetorial-estrutural Ψ^{vst}, entretanto o domínio dos termos no modelo Ψ^{vstt} é maior, pois agora os termos são indexados levando em consideração o seu tipo. A função similaridade é a mesma, ou seja, é dada pelo produto interno entre os vetores consulta e documento.

O modelo Ψ^{vstt} não é viável em termo de implementação, pois o número de termos a serem indexados é muito grande. Utilizando o modelo *sTerm* este problema é contornado, pois a indexação é feita em tempo de consulta, a qual faz a poda da árvore levando em consideração o tipo da consulta.

Capítulo 6

Conclusões e Trabalhos Futuros

6.1 Conclusões

A estrutura funcional baseada em λ-cálculo fornece uma ferramenta para representar, comparar, combinar e construir modelos de RI. Além disso, define um nível de abstração maior que os modelos de RI tradicionais e menor que outros meta-modelos genéricos como Caracterização BR-Formal. Isso permite trabalhar com aplicações teóricas e práticas, tornando-o prático no sentido de implementação e não tão genérico. Este *framework* é caracterizado por sua generalidade e pelo formalismo descrito através da notação do λ-cálculo para definição das funções.

O *framework* proposto tem como elemento central da representação a notação de funções. Esta estratégia difere dos modelos tradicionais, que geralmente consideram pesos de termos, vetores, etc como fundamentos. A representação funcional é importante para estudar características e propriedades dos modelos de RI. Para tal, utilizamos a notação funcional λ-cálculo que identifica os argumentos e valores das funções presentes nos modelos. A passagem dos modelos de RI para a estrutura funcional possibilita a comparação e combinação entre eles, pois todos os modelos são representados utilizando a mesma notação funcional, o λ-cálculo. Sendo assim, podemos identificar as funções que definem os modelos e assim construir, comparar e combinar modelos de RI.

Neste trabalho, representamos vários modelos de RI na estrutura funcional. Dentre estes destacamos os modelos: vetorial, probabilístico, booleano, redes de crença, *sTerm* e o baseado em ontologia. Na tabela 6.1 apresentamos um resumo da representação destes modelos na estrutura funcional.

Na primeira coluna (Ψ) descrevemos o nome dos modelos funcionais. Na segunda coluna ($\mathbf{D}_f$) identificamos a lista dos termos funcionais que caracteriza os documentos funcionais de seus respectivos modelos. Por exemplo, na segunda coluna e segunda linha temos a lista $[peso_j]$ contendo apenas o termo funcional $peso_j$, isto significa que os documentos funcionais no modelo Ψ^v são representados pela lista $[peso_j]$. Na terceira coluna ($\mathbf{Q}_f$) identificamos a lista dos termos funcionais que caracteriza as consultas funcionais de seus respectivos modelos. Na quarta coluna (Δ), descrevemos a função similaridade do seu respectivo modelo funcional. Finalmente, na quinta coluna (Tipo), identificamos o tipo da função similaridade. Portanto, a partir da segunda linha, temos que cada linha descreve a representação de um modelo na estrutura funcional e o tipo de sua respectiva função similaridade.

Ψ	$\mathbf{D}_f$	$\mathbf{Q}_f$	Δ	Tipo
Ψ^v λ-vetorial	$[peso_j]$	$[peso_q]$	$\lambda f \lambda g. \frac{\sum_{i=1}^{t} f(k_i).g(k_i)}{\sqrt{\sum_{i=1}^{t} f(k_i)^2} \times \sqrt{\sum_{i=1}^{t} g(k_i)^2}}$	Δ_{nrs}
Ψ^p λ-probabilístico	$[peso_j]$	$[peso_q$ $prob_q,$ $\overline{prob}_q]$	$\lambda f \lambda g \lambda h \lambda h_1. \sum_{i=1}^{t} f(k_i) \times g(k_i) \times (\frac{h(k_i)}{1-h(k_i)} + \frac{1-h_1(k_i)}{h_1(k_i)})$	$*$ Δ_{nr}
Ψ^b λ-booleano	$[bol_j^b]$	$[bol_q^b]$	$\lambda f \lambda g.((f(k_1, \ldots, k_t)\ implica\ g(k_1, \ldots, k_t)) \rightarrow 1 \mid 0)$	Δ_{nr}
Ψ^{rc} λ-redes de crença	$[peso_{e1_j},$ $\ldots,$ $peso_{ev_j}]$	$[peso_{e1_q},$ $\ldots,$ $peso_{ev_q}]$	$\lambda \vec{f} \lambda \vec{g}.((1-(1-R_{jq}) \cdot (1-f_2 \cdot g_2) \ldots (1-f_v \cdot g_v)))$	Δ_{nrs}
Ψ^{st} λ-sterm	$[peso_j^{st},$ $tipo]$	$[peso_q^{st},$ $tipo_q]$	$\lambda f \lambda g \lambda h \lambda h_1.((g = h_1) \rightarrow \frac{\sum_{i=1}^{m} f(T_i).h(T_i)}{\sqrt{\sum_{i=1}^{m} f(T_i)^2} \times \sqrt{\sum_{i=1}^{m} h(T_i)^2}} \mid 0)$	Δ_{nr}
Ψ^{ows} λ-ontologia	$[pesoSem_j,$ $relEqui,$ $indexSem]$	$[pesoSem_q]$	$\lambda f \lambda g. \frac{1}{\sqrt{\sum_{i=1}^{t} (f(X_i)-g(X_i))^2}}$	Δ_s

Tab. 6.1: Tabela de representação dos modelos funcionais.

($*$) Se considerarmos a constante de normalização, temos que a função similaridade do modelo funcional Ψ^p é normalizada e reflexiva, portanto Δ_{nr}.

Tendo em vista a tabela 6.1, observamos que a representação dos modelos de RI na estrutura funcional usando a notação λ-cálculo, permite a identificação e caracterização das funções presentes nos modelos de RI, facilitando o estudo das propriedades e semânticas destes modelos.

Além disso, a estrutura funcional pode ser usada para generalizar modelos de RI que podem

ser expressos por um algoritmo, pois ela é baseada em funções. Uma de nossas contribuições é a proposta de uma estrutura funcional baseado em λ-cálculo capaz de formular novos modelos, bem como permitir a combinação entre alguns modelos usando funções.

Outra vantagem é a proposta de uma metodologia para realizar a comparação entre modelos de RI, algebricamente, sem a necessidade de realizar experimentos. Nosso trabalho define uma formalização, baseada na notação λ-cálculo, para comparação de equivalência entre modelos. A comparação entre modelos é importante devido às seguintes razões: para um melhor entendimento do relacionamento entre os modelos comparados, para reutilização de código ou implementação de um modelo e para um melhor entendimento da semântica de similaridade. Neste trabalho, utilizando a estrutura funcional construímos modelos vetoriais alternativos equivalentes a modelos existentes. Esta mesma estratégia de análise também pode ser feita considerando outros modelos, diferentes daqueles aqui considerados.

6.2 Trabalhos Futuros

Os trabalhos futuros incluem os seguintes tópicos:

- Estender a representação na estrutura funcional para outros modelos, por exemplo, modelos baseados em lógica, redes neurais ou Inteligência Artificial.

- A realização de experimentos para comparar modelos que foram identificados como não equivalentes por meio da estrutura funcional.

- Tendo em vista o uso da notação λ-cálculo para definição das funções presentes nos modelos, podemos pensar na utilização da semântica denotacional para especificarmos os modelos. Com isso, podemos obter uma descrição mais precisa da semântica dos modelos e além disso, o uso da semântica denotacional facilita na implementação e desenvolvimento de uma ferramenta para comparação experimental entre modelos de RI baseada nos conceitos da estrutura funcional.

- Utilizar semânticas de modelagem diferentes para construção de novos modelos equivalentes

aos modelos existentes, porém mais simples e de fácil implementação que os modelos existentes.

- Análise e comparação da complexidade dos algoritmos utilizados por diferentes modelos. Também, a análise da complexidade de gerenciamento das funções envolvidas em cada modelo, utilizando lógica tipada.
- Quais condições um modelo baseado em peso de termos no documento deve satisfazer para ser representado de forma equivalente em um modelo vetorial.

Referências Bibliográficas

[1] Abinader Júnior, E.M. **Combinação e avaliação de múltiplas fontes de evidências para recuperação de documento na web**. Dissertação (Mestrado em Ciência da Computação), Universidade Federal do Amazonas, Instituto de Ciências Exatas, Amazonas, Manaus, 2004.

[2] Baader, F.; McGuinness, D.; Nardi, D. **The Description Logic Handbook: Theory, Implementation and Applications**. Cambridge, UK: Cambridge Univ.Press,2003.

[3] Baeza-Yates, R.; Ribeiro-Neto, B. **Modern Information Retrieval**. Addison Wesley, 1999.

[4] Barendregt, H. **The Lambda Calculus - Its Syntax and Semantics**. North-Holland, Amsterdam, 1981.

[5] Berners-Lee, T.; Lassila, O.; Hendler, J. **The Semantic Web**. Scientific American, 284 (5), pp. 34-43, 2001.

[6] Bordogna, G.; Pasi, G. **Linguistic aggregation operators in fuzzy information retrieval**. *International Journal of Intelligent Systems*,pg.233-248,1995.

[7] Breitman, K.K. **Web semântica: a internet do futuro**. Rio de Janeiro, LTC, 2005.

[8] Bruza, D.P.; Lalmas, M. **Logic based information retrieval: Is it really worth it?** In *Proceedings of WIRUL 96, the SecondWorkshop on Information Retrieval, Uncertainty and Logic(Glasgow)*. 1996.

[9] Bruza, D. P.; Crestani, F.; Lalmas, M. **Second workshop on logical and uncertainty models for information systems**. In *Proceedings of DEXA*. IEEE Press, 2000.

[10] Carmel, D. et al. **An extension of the Vector Space Model for Querying XML Documents via XML fragments.** In *ACM SIGIR*, Finland,2002. Workshop on XML and IR, Finland: ACM, 2002.

[11] Cerqueira, A. G.; Silva, G.B.; Cardoso, O.N.P. **Uma visão geral de XML em Recuperação de Informação**. Revista eletrônica de iniciação científica,v.2,n.2, 2002.

[12] Chen, S.P. **On inference rules of logic-based information retrieval systems**. *Information Processing and Management*, pg. 43–59, 1994.

[13] Chiaramella, Y. **About retrieval models and logic**. *The Computer Journal*, pg. 233–241, 1992.

[14] Church, A. **A set of Postulates for the Foundation of Logic**. *Annals of Mathematics, second series, 33,pp. 346-366, 1932*.

[15] Church, A. **An unsolvable problem of elementary number theory**. *American Journal of Mathematics, 58,pp. 345-363, 1936*.

[16] Christopher, C.Y.; Nan Liu. **Measuring similarity of semi-structured documents with context weights**. *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, Seattle, Washington, USA, 2006.

[17] Croft, B.W. **Knowledge-based and statistical approaches to text retrieval**. *IEEE Expert: Intelligent Systems and Their Applications*, 8(2):8–12, 1993.

[18] Croft, B.W. **Effective text retrieval based on combining evidence from the corpus and users**. *IEEE Expert: Intelligent Systems and Their Applications*, 10(6):59–63, 1995.

[19] Dominich, S. **Formal foundation of information retrieval**. In *Proceedings of the Workshop on Mathematical/Formal Methods in Information Retrieval at the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pg. 8–15, Athens, Greece, 2000.

[20] Dominich, S. **A unified mathematical definition of classical information retrieval**. *Jornal of the American Society for Information Science*, 51(7):614–624, 2000.

[21] Dominich, S. **On applying formal grammar and languages, and deduction to information retrieval modelling**. In *of the Workshop on Mathematical/Formal Methods in Information Retrieval at the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pg. 37–41, New Orleans, USA, 2001.

[22] Frakes, W.B. & Baeza-Yates, R. **Information Retrieval and Data Structures**. Prentice Hall, 1992.

[23] Fuhr, N. **Language models and uncertain inference in information retrieval**. In *Proceedings of the Language Modeling and IR workshop*.

[24] Fuhr, N. **Probabilistic models in information retrieval**. *The Computer Journal*, 35(3):243–255, 1992.

[25] Gordon, M.J.C. **The Denotational Description of Programming Languages: An Introducion**. Springer-Verlag, New York, NJ, USA, 1979.

[26] Gordon, M.J.C. **Programming Language Theory and Its Implementation**. Prentice-Hall, NJ, USA, 1988.

[27] Huibers, C.W.T; Bruza, D.P. **Situations: A general framework for studying Information Retrieval**. In R. Leon, editor, *Information retrieval: New systems and current research, Proceedings of the 16th Research Colloquium of the British Computer Society Information Retrieval Specialists Group*, pg. 3–25. Taylor Graham, Drymen, Scotland, 1996.

[28] Horrocks, I. **Using an expressive description logic: FACT or fiction?**. *Proceedings of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning*, pp. 636-647, Morgan Kaufmann, Trento,Italy, 1998.

[29] Jang, H.; Kim, Y.; Shin, D. **An Effective Mechanism for Index Update in Structured Documents**. *In: ACM Conference on Information and Knowledge (CIKM)*, p. 383–390, Kansas City, Missouri 1999.

[30] Jun-feng, S. et al. **Ontology-Based Information Retrieval Model for the Semantic Web**. *IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05)*, pp.152-155, 2005.

[31] Kang, H.; Choi, K. **Two-level document ranking using mutual information in natural language information retrieval**. *Inf. Process. Manage.*, 33(3):289–306, 1997.

[32] Kleene, S. **A theory of positive integers in formal logic**. *American Journal of Mathematics*, 57, pp. 153-173 and 219-244, 1935.

[33] Kleinberg, J.M. **Authoritative sources in a hyperlinked environment**. *In Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pg. 668-677, 1998.

[34] Lafferty, J.; Zhai, C. **Probabilistic relevance models based on document and query generation**. In *W. B. Croft and J. Lafferty, editors, Language Modeling and Information Retrieval*. Kluwer Academic Publishers, 2003.

[35] Lalmas, M; Bruza, D.P. **The use of logic in information retrieval modeling**. *Knowledge Engineering Review. In press.*, 13(3):263–295, 1998.

[36] McLennan, B. J. **Functional Programming: Pratice and Theory**. Addison-Wesley,Reading,Mass. 1990.

[37] Montejo, R. A. **Formal models for IR: a review and a proposal for keyword assignment**. In *Workshop on Mathematical/Formal Methods in Information Retrieval*. ACM-SIGIR, 2003.

[38] Nie, Y. J. **Un Modèle de Logique Générale pour les Systemes de Recherche d'Informations. Application au Prototype RIME**. PhD thesis, Université Joseph Fourrier, Grenoble, France, 1990.

[39] Oliveira, L.C. **Meta-Modelo Funcional Para Recuperação de Informação**. Dissertação (Mestrado em Ciência da Computação), Universidade Federal de Uberlândia, Faculdade de Computação, Uberlândia, Minas Gerais, 2006.

[40] Pearl, J. **Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference**. Morgan Kaufmann Publishers, Inc., 1988.

[41] Pinheiro de Cristo, A.M. et al. **Bayesian belief networks for IR**. *International Journal of Approximate Reasoning*, 34(2-3):163–179, 2003.

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

[43] Robertson, S.E.; Sparck Jones, K. **Relevance weighting of search terms**. In *Journal of the American Society for Information Sciences*, v.27, n.3, p.129-146, 1976.

[44] Salton, G.; Wong, A.; Yang, C.S. **A vector space model for automatic indexing**. Technical Report TR74-218, Cornell University, Computer Science Department, 1974.

[45] Salton, G.; Buckley, C. **Term weighting approaches in automatic text retrieval**. *Technical Report TR87-881, Department of Computer Science, Cornell University, 1987. Information Processing and Management, Vol.32(4), 1996,p.431-443*.

[46] Salton, G.; Yang, C.S.; Yu, C.T. **A theory of term importance in automatic text analysis**. *Journal of the American Society for Information Science*, pg. 33–44, Jan-Feb 1975.

[47] Schlieder, T.; Meuss, H. **Querying and Ranking XML Documents**. In *Journal of the American Society for Information Systems*, v.53, n.6, p.489-503, Apr. 2002.

[48] Silva, I. et al. **Link-based and content-based evidential information in a belief network model**. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pg. 96–103, Athens, Greece, July 2000.

[49] Souza, J.N. **Lógica para Ciência da Computação: fundamentos da linguagem, semântica e sistemas de dedução**. Elsevier, Rio de Janeiro, 2002.

[50] Turtle, H.; Croft, B. W. **Evaluation of an inference network-based retrieval model**. *ACM Transactions on Information Systems*, 9(3):187–222, July 1991.

[51] van Rijsbergen, J.C. **Information Retrieval**. Butterwords, 1979.

[52] van Rijsbergen, J.C. **A non-classical logic for information retrieval**. *The Computer Journal*, 29(6), 1986.

[53] Wang, X.L. et al. **Enhancive Index for Structured Document Retrieval**. *In: International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems (RIDE)*, 12,p. 256-264, Califórnia, 2002.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo