Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Computação e Automação Industrial

Utilização de funções LSH para busca conceitual baseada em ontologias

Autor: Luciano Bernardes de Paula
Orientador: Prof. Dr. Maurício Ferreira Magalhães

Tese de Doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Doutor em Engenharia Elétrica. Área de concentração: **Engenharia de Computação**.

Banca Examinadora

Prof. Dr. Maurício Ferreira Magalhães (Presidente) ... DCA/FEEC/Unicamp
Prof. Dr. Luis Carlos Trevelin ..... DC/UFSCar
Prof. Dr. Marcello Peixoto Bax ..... ECI/UFMG
Prof. Dr. Edmundo Roberto Mauro Madeira ..... IC/Unicamp
Prof. Dr. Juan Manuel Adán Coello ..... PUC/Campinas

Campinas, SP

Julho/2011

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Paula, Luciano Bernardes de
P281u Utilização de funções LSH para busca conceitual baseada em ontologias / Luciano Bernardes de Paula. – Campinas, SP: [s.n.], 2011.

Orientador: Maurício Ferreira Magalhães.
Tese de Doutorado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Web Semântica. 2. Ontologias (Recuperação da informação). 3. Hashing (Computação). I. Magalhães, Maurício Ferreira. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês:	Use of LSH functions for conceptual search based on ontologies
Palavras-chave em Inglês:	Semantic Web, Ontologies (Information retrieval), Hashing (Computer Science)
Área de concentração:	Engenharia de Computação
Titulação:	Doutor em Engenharia Elétrica
Banca Examinadora:	Edmundo Roberto Mauro Madeira, Juan Manuel Adán Coello, Luis Carlos Trevelin, Marcello Peixoto Bax
Data da defesa:	28/07/2011
Programa de Pós Graduação:	Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE DOUTORADO

Candidato: Luciano Bernardes de Paula

Data da Defesa: 28 de julho de 2011

Título da Tese: "Utilização de funções LSH para busca conceitual baseada em ontologias"

Prof. Dr. Maurício Ferreira Magalhães (Presidente): _____

Prof. Dr. Juan Manuel Adan Coello: _____

Prof. Dr. Luis Carlos Trevelin: _____

Prof. Dr. Marcello Peixoto Bax: _____

Prof. Dr. Edmundo Roberto Mauro Madeira: _____

Resumo

O volume de dados disponíveis na WWW aumenta a cada dia. Com o surgimento da Web Semântica, os dados passaram a ter uma representação do seu significado, ou seja, serem classificados em um conceito de um domínio de conhecimento, tal domínio geralmente definido por uma ontologia. Essa representação, apoiada em todo o ferramental criado para a Web Semântica, propicia a busca conceitual. Nesse tipo de busca, o objetivo não é a recuperação de um dado específico, mas dados, de diversos tipos, classificados em um conceito de um domínio de conhecimento. Utilizando um índice de similaridade, é possível a recuperação de dados referentes a outros conceitos do mesmo domínio, aumentando a abrangência da busca. A indexação distribuída desses dados pode fazer com que uma busca conceitual por similaridade se torne muito custosa. Existem várias estruturas de indexação distribuída, como as redes P2P, que são empregadas na distribuição e compartilhamento de grandes volumes de dados. Esta tese propõe a utilização de funções LSH na indexação de conceitos de um domínio, definido por uma ontologia, mantendo a similaridade entre eles. Dessa forma, conceitos similares são armazenados próximos um dos outros, tal conceito de proximidade medida em alguma métrica, facilitando a busca conceitual por similaridade.

Palavras-chave: Web Semântica, similaridade, funções LSH, redes P2P.

Abstract

The volume of data available in the WWW increases every day. The Semantic Web emerged, giving a representation of the meaning of data, being classified in a concept of a knowledge domain, which is generally defined using an ontology. This representation, based in all the tools created for the Semantic Web, possibilitates the conceptual search. In this type of search, the goal is not to retrieve a specific piece of data, but several data, of several types, classified in a concept of an ontology. Using a similarity level, the retrieval of data that refer to other concepts of the domain is also possible, making the search broader. The distributed indexing of all these data may turn the conceptual search costly. The Internet holds several structures of distributed indexing, such as P2P networks, which are used in the distribution and sharing of huge volumes of data. This thesis presents how it is possible to use LSH functions to generate identifiers to concepts of a domain, defined using an ontology, keeping their similarity. This way, similar concepts are stored near each other, such distance measured in some metric, turning the conceptual search by similarity easier.

Keywords: Semantic Web, similarity, LSH functions, P2P networks.

Para os meus pais.

Agradecimentos

Em primeiro lugar, agradeço a Deus por ter me guiado até aqui.

Agradeço ao meu orientador, Prof. Dr. Maurício F. Magalhães, por todo apoio e amizade. Sem seus conselhos e sua experiência, este trabalho não teria se concretizado.

Agradeço aos meus pais, Ricardo e Raquel, e meus irmãos, Flavio e Henrique, por me apoiarem e me animarem nos momentos difíceis.

Agradeço à Márcia, por todo carinho e amor demonstrados todos esses anos. Ela sempre soube me ouvir quando era preciso que eu falasse e falar quando era preciso que eu ouvisse.

Agradeço aos meus amigos do LCA/DCA/Unicamp, por todos esses anos de amizade que, com certeza, continuará por muito tempo.

Agradeço aos meus amigos da graduação (Unesp/São José do Rio Preto) e do mestrado (UFSCar), por estarem sempre presentes, mesmo quando distantes.

Sumário

Lista de Figuras

Lista de Tabelas

Glossário

CBIR *Content-Based Image Retrieval*

CERN *Conseil Européen pour la Recherche Nucléaire*

DHT *Distributed Hash Table*

eTVSM *Enhanced Topic-based Vector Space Model*

FCA *Formal Concept Analysis*

HTML *HyperText Markup Language*

IA Inteligência Artificial

IP *Internet Protocol*

LSH *Locality Sensitive Hash*

MD5 *Message-Digest algorithm 5*

OAV Objeto-atributo-valor

OWL *Web Ontology Language*

P2P *Peer-to-peer*

RDF *Resource Description Framework*

RDFS *Resource Description Schema*

RHH *Random Hyperplane Hash*

SHA *Secure Hash Algorithm*

SIOC *Semantically-Interlinked Online Communities*

SON *Semantic Overlay Network*

TCP *Transmission Control Protocol*

URI *Uniform Resource Identifier*

W3C *World Wide Web Consortium*

WWW *World Wide Web*

XML *Extensible Markup Language*

XMLS *XML Schema*

Trabalhos Publicados Pelo Autor

Relacionados com esta tese:

1. L. B. de Paula, R. S. Villaça, M. F. Magalhães. “Analysis of concept similarity methods applied to an LSH function”. *Computer Software and Applications Conference 2011* (Compsac 2011), Munique, Alemanha, Julho de 2011.

2. L. B. de Paula, R. S. Villaça, M. F. Magalhães. “Organização de dados multimídia para busca conceitual baseada em ontologias”. *Simpósio Brasileiro de Sistemas Multimídia e Web* (WebMedia 2010), Belo Horizonte, Minas Gerais, Brasil, Outubro de 2010. *Menção honrosa de Melhor Artigo do WebMedia 2010*.

3. L. B. de Paula, R. S. Villaça, M. F. Magalhães. “A Locality Sensitive Hashing Approach for Conceptual Classification”. *4th International Conference on Semantic Computing* (ICSC 2010), Pittsburgh, EUA, Setembro de 2010.

4. L. B. de Paula, R. S. Villaça, M. F. Magalhães. “Uma proposta de função LSH baseada em ontologia para a indexação e recuperação de dados conceitualmente relacionados em redes P2P”. *3o. Seminário de Pesquisa em Ontologia no Brasil* (Ontobras 2010), Florianópolis, Santa Catarina, Brasil, Agosto de 2010.

5. R. S. Villaça, L. B. de Paula, M. F. Magalhães. “Avaliação de redes P2P baseadas no fenômeno Small World para o compartilhamento de conteúdos similares gerados por funções LSH”. *6th Workshop on Dynamic Networks and Peer-to-Peer Systems* (WP2P 2010), Gramado, Rio Grande do Sul, Brasil, Maio de 2010.

Outros artigos publicados durante o doutorado:

1. R. B. Freitas, L. B. de Paula, E. Madeira, F. L. Verdi. “Using virtual topologies to manage inter-domain QoS in next-generation networks”. *International Journal of Network Management*, volume 20, edição 3, páginas 111-128 , Maio/Junho de 2010.

2. R. Pasquini, L. B. de Paula, F. L. Verdi, M. F. Magalhães. “Domain Identifiers in a Next Generation Internet Architecture”. *IEEE Wireless Communications & Networking Conference* (WCNC 2009), Budapeste, Hungria, Abril de 2009.

3. L. B. de Paula, R. S. Villaça, F. L. Verdi, M. F. Magalhães. “A Web Service-based Network Composition Architecture for the Next Generation Internet”. *5th International Workshop on Next Generation Networking Middleware* (NGNM 2008), Samos, Grécia, Setembro de 2008.

4. W. Wong, R. S. Villaça, L. B. de Paula, R. Pasquini, F. L. Verdi, M. F. Magalhães. “An Architecture for Mobility Support in a Next Generation Internet”. *The 22nd IEEE International Conference on Advanced Information, Networking and Applications* (AINA 2008), Okinawa, Japão, Março de 2008.

5. W. Wong, R. Pasquini, R. S. Villaça, L. B. de Paula, F. L. Verdi, M. F. Magalhães. "A Framework for Mobility and Flat Addressing in Heterogeneous Domains". *25o. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos* (SBRC 2007), Belém, Pará, Brasil, Maio de 2007.

Capítulo 1

Introdução

A maneira com a qual o ser humano adquire e processa informações, ou seja, constrói um conhecimento, vem sendo amplamente estudada em diversas áreas, tais como a Filosofia e a Psicologia. Em várias áreas da Ciência da Computação, principalmente na Inteligência Artificial, busca-se as melhores maneiras de se representar esse conhecimento, de forma que seja possível fazer com que máquinas o processem, o interpretem e o utilizem. Ao se deparar com um dado, seja em forma de texto, de imagem, ou de um *website*, um ser humano, de maneira relativamente fácil e rápida, consegue entender o significado deste, com o auxílio de informações previamente existentes em seu cérebro. O mesmo não pode ser dito das máquinas. São necessários formalismos, anotações e procedimentos explícitos que façam com que as máquinas consigam adquirir, processar e utilizar informações, caso contrário essas informações não possuem nenhuma utilidade em um sistema computacional.

Um conceito importante, relacionado com o processamento de informações e utilizado pelos seres humanos, é o da similaridade. É a partir da semelhança que novas situações tenham com situações previamente conhecidas ou vividas que os seres humanos, de frente a novos problemas ou novas informações, conseguem resolvê-los e entendê-las. É utilizando o conceito de similaridade que o ser humano consegue classificar objetos, formar conceitos e fazer generalizações. A noção de similaridade é fundamental em áreas da Psicologia Cognitiva [3][4][5], aquisição de conhecimento [6], gerenciamento de dados e organização de informação [7][8][9].

Uma forma de se definir um domínio de conhecimento, ou seja, suas entidades e atributos, e as relações entre essas entidades com outras do mesmo domínio, é utilizando uma ontologia. Pela definição de um domínio feita em forma de uma ontologia, é possível a inferência de novas informações sobre aquele domínio, mesmo que essas não estejam definidas de forma explícita. Em Ciência da Computação, ontologias são usadas como modelos de dados que definem entidades, ou conceitos, e as regras que as relacionam, geralmente representadas por grafos ou redes semânticas, consistindo em hierarquias de conceitos. Em um sistema computacional, a definição de um domínio por uma

ontologia possibilita a determinação da similaridade entre os conceitos desta. Várias são as técnicas desenvolvidas para tal determinação, cada uma levando em consideração certas características da ontologia, como o tamanho do caminho entre conceitos, a profundidade desses na hierarquia, entre outros [10][11].

A definição de domínios de conhecimento, entidades e relações é especialmente importante no cenário da Web Semântica (*Semantic Web*). Em 2001, Tim Berners-Lee, inventor da WWW (*World Wide Web*), apresentou em [12] como é possível evoluir a sua invenção por meio de uma camada que dá significado aos dados. Dessa forma, há uma evolução de uma rede de documentos, como a atual WWW, para uma rede de coisas, na qual tudo pode ter uma representação computacional e essa representação passa a ter um significado para as máquinas, geralmente definido por um conceito dentro de um domínio. Consequentemente, o conceito de similaridade sobre os dados da Web Semântica torna-se a cada dia mais importante, abrindo novas opções de tratamento e processamento dos dados pelo seu significado conceitual.

A forma mais comum do uso de similaridade para a comparação de dados, encontrada na literatura, é feita comparando dois ou mais dados pelo seu conteúdo. Ou seja, textos são comparados pelas suas palavras-chave, imagens por suas características visuais, etc. Por outro lado, a partir da classificação de um dado em um domínio de conhecimento, esta tese explora o fato de que é possível a comparação de dados no nível conceitual, ou seja, duas imagens visualmente diferentes podem ser consideradas similares, do ponto de vista conceitual, por retratarem coisas da mesma natureza. Por exemplo, uma foto de um cachorro da raça São Bernardo é diferente de uma foto de um cachorro da raça Dachshund, mas, ao mesmo tempo, podem ser classificadas em um mesmo conceito, já que ambas representam raças de uma mesma espécie de animal. Dois textos podem falar do mesmo conceito, ou seja, de um mesmo assunto, mas não utilizar as mesmas palavras-chave. No nível conceitual é possível, inclusive, a comparação de dados considerados incomparáveis como, por exemplo, uma imagem e um texto. A Figura 1.1 mostra essa relação entre os dados.

A utilização de conceitos para descrever dados permite a *busca conceitual* [13][14][15]. Nesse tipo de busca, não recupera-se um objeto específico, mas informações ou dados sobre um determinado conceito. Por exemplo, seguindo a Figura 1.1, ao se buscar informações sobre o conceito F seriam retornados todos os dados, de qualquer tipo, classificados e indexados sob aquele conceito. Podem, até mesmo, ser recuperados os localizadores de bases de dados que contenham dados classificados sob o conceito buscado. É importante notar que um mesmo dado pode ser visto de várias maneiras, dependendo do domínio de conhecimento em que esse está sendo classificado. Por exemplo, na Figura 1.1, F pode representar o conceito *Cão* em um domínio de conhecimento que defina *Mamíferos* ou o conceito *Animal de estimação* em um domínio de conhecimento que defina *Coisas encontradas em uma casa*.

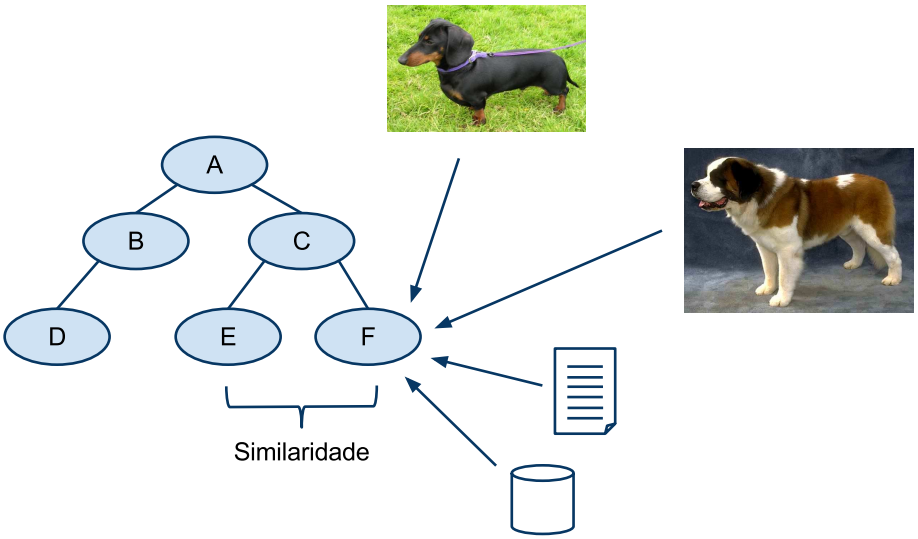


Fig. 1.1: A partir de uma classificação definida por uma ontologia de domínio, diversos dados de vários tipos (imagens, textos, etc) podem ser relacionados entre si.

Uma maneira de aumentar o escopo de uma busca conceitual é utilizar a similaridade entre conceitos, definida por alguma função de similaridade. Como pode ser visto na Figura 1.1, os conceitos da ontologia que representa um domínio possuem certa similaridade entre si, sendo alguns conceitos mais, ou menos, similares a outros. Em uma busca conceitual por similaridade, além das informações classificadas pelo conceito buscado, outras, classificadas em conceitos similares, também podem ser recuperadas, aumentando a abrangência da busca. A similaridade entre conceitos pode, inclusive, ser utilizada como fator de ordenação dos resultados (*ranking*), de acordo com o grau de similaridade dos conceitos em relação ao conceito buscado.

Um ponto importante em tudo o que foi apresentado até agora é a necessidade de organização desses dados conceitualmente classificados, de forma a facilitar a busca por dados relacionados. Devido ao grande volume de dados presentes atualmente na WWW, a indexação se torna cada vez mais descentralizada, o que resulta em situações nas quais servidores que disponibilizam dados e serviços similares podem estar distantes uns dos outros na WWW. Em [16] é apresentado o conceito de portais de conhecimento. Basicamente, portais de conhecimento são portais Web com pouco conteúdo próprio, mas que fornecem inúmeros *links* apontando para diversos outros servidores que possuam as informações desejadas. Por trás desses portais, existe a necessidade de se indexar e compartilhar um grande volume de dados, eventualmente classificados semanticamente.

Nesse contexto da organização dos dados, uma estrutura de rede comumente utilizada na indexação e compartilhamento de grandes volumes de dados são as tabelas *hash* distribuídas (*Distributed Hash Tables - DHT*)[17]. Basicamente, as DHTs são formadas por nós distribuídos, organizados em

uma rede sobreposta[1] (*overlay*) com uma estratégia de roteamento implementada entre eles. Vários modelos de rede *peer-to-peer* (P2P) são implementados sobre DHTs. Comumente, a identificação de um dado em uma DHT, ou rede P2P, é feita por um valor gerado por uma função de *hash*, tais como MD5 (*Message-Digest algorithm 5*) e os algoritmos da família SHA (*Secure Hash Algorithm*), como o SHA-1. Essas funções têm como característica a sua aleatoriedade na geração dos valores de *hash* resultantes a partir de um certo dado. Ou seja, os valores gerados não guardam qualquer relação semântica que porventura exista entre os dados inseridos na função de *hash*. Isso implica em desafios em uma eventual busca por similaridade. Se dois dados, similares por conceito ou conteúdo, forem identificados e indexados em uma rede P2P por chaves criadas por funções de *hash* tradicionais, as posições que as representações desses dados ocuparão no espaço de indexação da rede sobreposta não terão relação alguma, podendo inclusive ser indexados "longe" uns dos outros no espaço virtual. Isso gera maior esforço na recuperação de dados considerados similares.

Uma alternativa utilizada na indexação de dados, sem perder a similaridade entre esses, é a utilização de funções *hash* sensíveis à localidade (*Locality Sensitive Hash - LSH*). Esse tipo de função tem como característica principal a criação de valores de *hash* que mantenham, de maneira probabilística, a similaridade entre dois dados. Basicamente, para dois dados que possuam similaridade *p* determinada por uma função de similaridade, uma função LSH tem probabilidade *p* de gerar valores de *hash* idênticos para esses dois dados. A utilização de tais valores de *hash*, gerados por funções LSH, na indexação dos dados, faz com que esses sejam armazenados próximos uns dos outros no espaço de indexação, sendo a proximidade medida em alguma métrica. As funções LSH são sempre atreladas a uma função de similaridade e um tipo de representação dos dados, como, por exemplo, a função das permutações independentes *Min-Wise* [18][19][20], que foi criada para dados representados por conjuntos de inteiros e que segue a similaridade de Jaccard, e a *Random Hyperplane Hash* (RHH) [21], criada para dados representados por vetores, seguindo a similaridade do cosseno. Ambas serão detalhadas no Capítulo 4.

Na literatura é comum encontrar trabalhos que utilizam funções LSH para comparar e relacionar dados do ponto de vista dos seus conteúdos, ou seja, não há relação conceitual determinada a priori entre os dados. Esta tese tem como um dos seus objetivos principais apresentar a criação de uma família de funções LSH que gere valores de *hash* para conceitos de uma ontologia, mantendo assim a similaridade entre eles. Em outras palavras, pode ser dito que a intenção é a representação de um conhecimento estabelecido por humanos, como no caso uma ontologia de domínio, em um valor binário usado na indexação de dados classificados nesse domínio de conhecimento. A Figura 1.2 ilustra esse cenário.

Esta tese apresenta a criação de identificadores para conceitos de uma mesma ontologia de do-

[1] Redes sobrepostas, também chamadas de redes *overlay*, são redes lógicas sobre redes físicas.

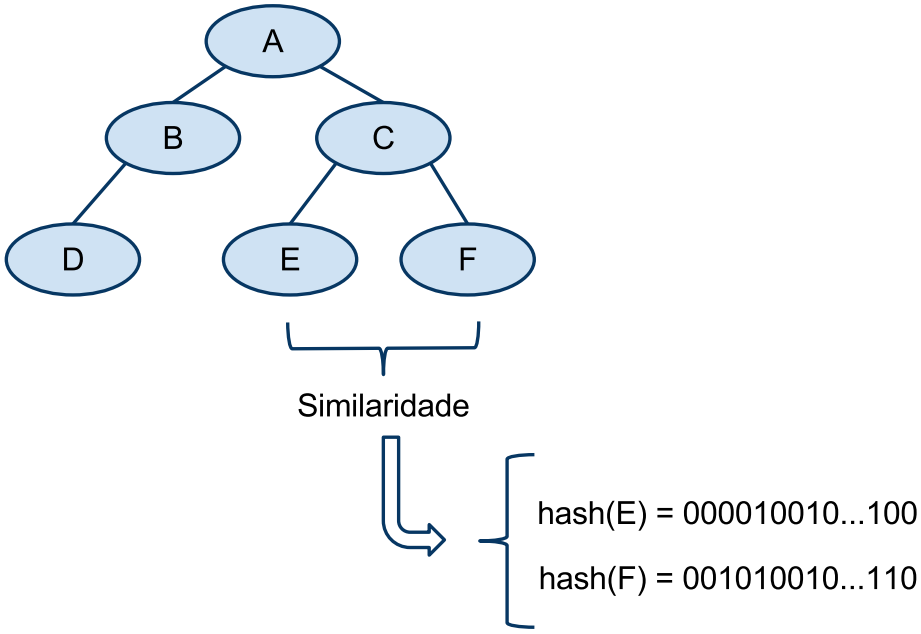


Fig. 1.2: Valores de *hash* são criados, para cada conceito, de forma que mantenham a similaridade entre eles, por exemplo na distância de Hamming entre os identificadores, que contabiliza o número de bits diferentes entre duas cadeias binárias.

mínio utilizando funções LSH e tendo como objetivo facilitar a busca conceitual por similaridade em um sistema de indexação distribuído. Dessa forma, conceitos similares possuirão identificadores similares e serão indexados próximos uns dos outros no espaço de indexação, facilitando a busca por conceitos similares.

Nas próximas seções serão apresentados os trabalhos que se relacionam com os assuntos abordados nesta tese e, em seguida, são resumidas as contribuições geradas pelas propostas aqui apresentadas.

1.1 Trabalhos relacionados

Esta seção apresenta alguns dos trabalhos encontrados na literatura relacionados com os assuntos abordados nesta tese. Os trabalhos citados foram divididos em três tipos, cada um deles representando uma das seguintes categorias:

- utilização de funções LSH (Seção 1.1.1) - trabalhos que utilizam funções LSH para relacionar dados comparáveis uns com os outros;
- utilização de ontologias e busca conceitual (Seção 1.1.2) - trabalhos que utilizam ontologias para organizar domínios de conhecimento e outros que, além disso, realizam a busca conceitual.

A busca conceitual baseada em ontologias faz parte do foco principal desta tese;

- indexação semântica de dados em redes P2P (Seção 1.1.3) - trabalhos que mostram a indexação de dados em redes P2P que utilizam a similaridade semântica, geralmente baseada em funções LSH.

Nas próximas subseções, serão apresentados cada um desses tipos, descrevendo alguns trabalhos que os representam.

1.1.1 Utilização de funções LSH

Na literatura são encontrados vários trabalhos que utilizam funções LSH para relacionar e comparar dados similares, sendo essa similaridade definida por uma função de similaridade.

Em [22], os autores apresentam como utilizar a função LSH *Min-Wise* (Seção 4.2.1) para relacionar imagens. Nesse trabalho as imagens são classificadas por suas "palavras visuais" (objetos ou formas presentes na imagem) e essas são usadas como conjuntos de palavras, aplicados à função *Min-wise*. Imagens que possuam conjuntos similares de palavras visuais possuem maiores chances de resultarem o mesmo valor de *hash*.

Em [23] os autores utilizam a função LSH RHH (Seção 4.2.2) para indexar imagens de forma a manter a similaridade entre elas. Para representar as imagens de forma vetorial são utilizadas características dessas, como formas e tons de cores presentes na imagem. O trabalho visa problemas de classificação e *clustering* de grandes bases de dados.

Em [24], uma função LSH similar ao RHH é utilizada para relacionar imagens, representadas por vetores de características, tais como os tons de cores e tipos de textura, visando resolver o problema da busca por dados similares em um espaço de indexação. Os valores obtidos pelo *hash* são distribuídos em recipientes (*buckets*). Imagens similares possuem grande probabilidade de serem indexadas no mesmo recipiente, facilitando a busca por similaridade.

Em [25] é utilizada uma variação da função RHH para relacionar diversos tipos de dados, representados por vetores. Foram utilizados dados estatísticos sobre jogadores da NBA[2] e medidas de temperatura e umidade feitas por redes de sensores. A ideia é diminuir o tamanho das informações que representam os dados sem perder a similaridade entre eles (muito importante no cenário de redes de sensores).

Os autores de [26] utilizam funções LSH para encontrar, em uma base de dados, arquivos de áudio similares. Um usuário indica um trecho de uma música ou som que ele queira recuperar e são retornados aqueles arquivos que possuem sons similares ao que ele indicou. O usuário pode,

[2] Associação Nacional de Basquete dos EUA - *National Basketball Association*.

inclusive, utilizar um microfone e assoviar ou cantar um trecho da música buscada. Utilizando uma função LSH, todas as músicas indexadas na base de dados, que possuam trechos parecidos com o buscado, são retornadas.

Em [27], os autores utilizam funções LSH para identificar vídeos similares em uma base de dados. O intuito é identificar em uma grande base de dados, por exemplo o *Youtube*[3], vídeos que infrinjam termos de *copyright*, comparando os seus conteúdos com o conteúdo original.

Os autores de [28] criaram uma função LSH para buscas por similaridade para dados representados por vetores com um grande número de dimensões. Os testes são feitos em uma base de dados de imagens e outra de arquivos de áudio de conversas entre pessoas ao telefone, divididas em arquivos menores, delimitando cada palavra. O intuito do trabalho é melhorar a eficiência, em termos do espaço necessário para armazenar as tabelas *hash* que comparam os dados em relação à similaridade.

A utilização de funções LSH em sistemas CBIR (*Content-Based Image Retrieval*) é o foco em [29]. Cada imagem é representada com um vetor de 238 dimensões, cada coordenada representando uma característica da imagem. A base de dados testada tinha cerca de 5.000 imagens divididas em 50 categorias. A partir de um grupo de imagens como exemplo foram buscadas outras que possuem valores de *hash* semelhantes gerados pela função LSH e é medida a precisão e a cobertura dos resultados obtidos.

Uma variação da função RHH é apresentada em [30], concebida para dados estatísticos, definidos por funções núcleo[4] (*kernel functions*). Os testes usaram uma base de imagens de objetos e fotos de vários pontos turísticos, retirados de [31], mas o método aplicado também pode ser usado para vários tipos de dados.

1.1.2 Utilização de ontologias simples e busca conceitual

A busca conceitual é explorada em vários trabalhos presentes na literatura. Nesse tipo de busca, o objetivo é recuperar dados dados classificados sob um conceito e não somente um dado específico. Em seguida serão citados alguns deles, com um breve comentário sobre cada um.

Em [13], os autores apresentam a ideia da busca conceitual, mostrando as vantagens que essa possui sobre buscas baseadas em palavras-chave. O trabalho aborda a categorização de dados *on the fly*[5]. Alguns testes relacionados com a categorização automática de dados são apresentados.

A autora de [15] discute e apresenta uma maneira de se relacionar conceitos abordando a análise

[3] Youtube - http://www.youtube.com (acessado em 29 de março de 2011).

[4] Funções núcleo são funções usadas em técnicas de estimativas não-parametrizadas. Os núcleos são usados na estimativa da densidade de variáveis aleatórias ou na regressão de núcleo para estimar a expectativa condicional de uma variável aleatória.

[5] A expressão *on the fly* refere-se a algum processo que é feito sem tratamento prévio, sendo processado durante o funcionamento de um sistema ou durante a utilização dos dados.

formal de conceitos (*Formal Concept Analysis* - FCA) [32][33]. É dito que o propósito da FCA é dar suporte ao usuário analisando e estruturando um domínio de interesse. Dado um domínio, um conceito em FCA é um par de conjuntos: um conjunto de objetos, que são instâncias de um conceito daquele domínio, e um conjunto de atributos que descrevem o conceito.

Em [14] são descritos os avanços feitos pelo *Institutional Repository Search Project*[6] em pesquisas sobre a recuperação de informações em repositórios. No artigo, os autores afirmam que informações armazenadas em repositórios ou grupo de repositórios só são úteis se puderem ser encontradas, transformadas e interpretadas para satisfazer a necessidade do usuário. O artigo apresenta as experiências e sucessos obtidos pelo projeto utilizando a busca conceitual.

O artigo [34] trata do processamento de grandes volumes de dados médicos. O trabalho utiliza FCA em grafos conceituais, organizando os dados pelo conceito em que esses são classificados dentro de um domínio, melhorando a busca por informações.

Em [35] é discutida a busca conceitual para textos referentes a leis. É citado que, nesse cenário, a recuperação de informação de forma conceitual depende da construção de uma representação viável do conhecimento. Na tentativa de se apresentar uma abordagem realista, o trabalho foi modelado levando-se em consideração a atividade de advogados e pesquisadores na recuperação de informações que necessitam. O trabalho apresenta como é possível construir um cenário de busca conceitual para esse tipo de informação, destacando suas vantagens.

O trabalho apresentado em [36] utiliza grafos conceituais para descrever o conteúdo de imagens. Dessa forma, facilita a busca por imagens similares, apresentando uma busca baseada em conceitos, levando em conta o conteúdo das imagens sem classificá-las dentro de um domínio de conhecimento. O trabalho mostra que a utilização de um modelo lógico para busca de informação não é apenas um modelo teórico, mas pode ser implementado para buscas eficientes.

Os autores de [37] tratam do problema da recuperação de informação em bases de dados mesmo quando a informação exata é inexistente. Para isso utilizam grafos conceituais como anotações intermediárias para a representação do conhecimento. O trabalho apresenta um algoritmo eficiente para comparação de grafos conceituais para inferência semântica.

O artigo [38] tem como foco a busca por códigos fonte similares. Esse tipo de busca é importante na identificação de duplicação de códigos e plágio. A duplicação de código é um problema conhecido em engenharia de *software* (cerca de 20% do total de códigos fonte de grandes sistemas são repetições [39]) e a identificação de plágio é algo importante quando envolve licenças comerciais e direitos autorais. A principal contribuição do trabalho é a representação de códigos fonte em um grafo conceitual, o que permite a comparação entre grafos para a identificação de similaridades.

Em [40], os autores afirmam que, em sistemas de busca, somente o uso de palavras-chave não é

[6] Encontrado em http://www.intute.ac.uk/irs (acessado em 28 de março de 2011).

suficiente, sendo necessária também a comparação conceitual. Utilizam conceitos organizados em ontologias e discutem formas de se comparar e utilizar a similaridade entre esses conceitos em buscas conceituais.

Os autores de [41] afirmam a importância da similaridade conceitual para o contexto da Web Semântica e propõem uma medida de similaridade para lógicas de descrição[7] complexas. Definem também um conjunto de critérios que uma medida como essa deve ter para que seja concordante com o comportamento semântico esperado.

O artigo [42] mostra como é possível aproximar a WWW, baseada em HTML (*HyperText Markup Language*), e a Web Semântica, baseada em RDF (*Resource Description Framework*), pela interligação de palavras em um texto e conceitos em uma ontologia. O artigo apresenta métodos automáticos de mapeamento de conceitos equivalentes importados de ontologias descritas em RDF, que permitem a integração de ontologias de domínios específicos utilizadas em recuperação de informação baseada em conceitos.

Em [43] é apresentada uma métrica para medir a similaridade semântica entre palavras. A abordagem é baseada em ontologias representadas por uma base de conhecimento geral para criar dinamicamente uma rede semântica baseada em propriedades linguísticas. Essa rede semântica é combinada com a métrica apresentada no trabalho para contabilizar a similaridade entre palavras, dessa forma obtendo uma estratégia eficiente para ordenar documentos digitais da Web, de acordo com a intenção de um usuário.

1.1.3 Indexação semântica em redes P2P

A indexação semântica de dados em sistemas distribuídos, mais precisamente em redes P2P, é abordada nos trabalhos citados a seguir.

Em [19] os autores propõem uma maneira de se resolver buscas por intervalo de valores em redes P2P. Uma busca por um intervalo de valores pode ser ilustrada como a recuperação de todos os registros, presentes em uma base de dados, de pessoas que possuam idade entre 30 e 50 anos. A abordagem utilizada foi indexar os dados levando em conta um intervalo fixo de valores. Por exemplo, para idade entre 30 e 50 anos, o conjunto de valores utilizados foi {30, ..., 50}. Esse conjunto de valores foi utilizado como um conjunto de inteiros, aplicando-se a função LSH *Min-wise*. Valores que estejam contidos nos mesmos intervalos possuem alta probabilidade de serem indexados com o mesmo identificador na rede P2P.

Em [44], os autores utilizam a função LSH *Min-wise* para indexar textos em uma rede P2P. Cada

[7] Lógica de descrição (*Description Logics*) é uma família de linguagens para representar um domínio de conhecimento, caracterizadas por semânticas formais bem definidas e um conjunto de operadores usados para definir relações existentes naquele domínio.

texto é representado por um conjunto de palavras-chave, e esse conjunto é utilizado pela função *Min-wise*, gerando um identificador semântico (chamado *semID*). Textos similares, ou seja, que possuam palavras-chave similares, possuem alta probabilidade de gerar o mesmo *semID*. Para aumentar a probabilidade de textos similares gerarem o mesmo *semID*, são utilizadas cerca de 20 funções do tipo *Min-wise* e cada *semID* criado é indexado em uma rede P2P. Dessa forma, aumenta-se a chance de que textos similares sejam indexados no mesmo nó da rede em pelo menos um desses *semIDs*.

Em [45] é utilizada a semântica dos dados para o roteamento de mensagens em uma rede P2P. Os nós classificam seus dados utilizando uma ontologia e divulgam os conceitos que possuem dados associados. Ao receber uma mensagem de busca, os nós a encaminham somente para os vizinhos que possuem dados do conceito procurado.

Ontologias também são utilizadas em [46] para agregar nós pela semântica dos seus conteúdos em uma rede *overlay* do tipo P2P, composta por várias SONs (*Semantic Overlay Networks*). Uma SON é composta por um grupo de nós que possuem dados com a mesma classificação semântica. Um nó escolhe de quais SONs ele fará parte de acordo com o tipo de dados que possui, podendo fazer parte de várias ao mesmo tempo. Buscas são encaminhadas para as SONs que possuem o tipo de dado procurado.

Em [47] é apresentado um modelo de rede P2P no qual, a partir de uma ontologia em comum, os nós podem escolher para qual vizinho enviar uma requisição de busca, baseados nos dados que esses vizinhos possuem. É utilizada uma função de similaridade entre conceitos da ontologia para o envio das mensagens, evitando-se assim mensagens em *broadcast* para todos os nós da rede.

Os autores de [48] organizam, em uma DHT, recursos e aplicações disponíveis em uma grade computacional (*grid*). Esses recursos e aplicações são descritos por ontologias que definem seus tipos e características. Cada domínio é indexado nos nós da DHT, facilitando a busca por recurso ou aplicação pelo respectivo tipo. É utilizada a similaridade entre conceitos de uma ontologia para a busca de recursos.

Uma DHT baseada em hipercubo e otimizada para buscas em *broadcast* é apresentada em [49]. O trabalho apresenta como construir tal rede, de forma a minimizar o número de mensagens necessárias para se atingir todos os nós que a compõem. O trabalho também cita a possibilidade de se construir essa DHT seguindo uma ontologia que defina os dados disponíveis na rede. A alternativa apresentada é a construção de vários hipercubos, cada um representando os nós que possuem dados de um certo conceito da ontologia.

1.2 Contribuições desta tese

A respeito dos trabalhos relacionados, apresentados nas seções anteriores, é importante citar alguns pontos que esses possuem em comum.

Os trabalhos relacionados, referentes à utilização das funções LSH, possuem como característica em comum a relação entre dados pelos seus conteúdos. Por exemplo, em [22], [23] e [24], imagens são representadas e comparadas pelo conteúdo (tons de cores, formas geométricas, etc) e não pelo que representam. Em [44], um trabalho que tanto usa funções LSH quanto indexa os resultados em uma DHT, textos são relacionados pelas palavras-chave presentes e não pelo assunto ou tema. Em [19] e [25], dados que possuem valores parecidos para determinada característica são indexados similarmente. Trechos de áudio e vídeos são comparados em [27] e [28], respectivamente. Esse tipo de abordagem na comparação dos dados gera dificuldades quando a busca é feita de maneira conceitual. Em nenhum desses trabalhos é possível uma busca do tipo: "recupere todos os vídeos de um jogo de futebol", "quais são as imagens que retratam raças de cães?" ou mesmo, "recupere vídeos, textos e imagens sobre música".

Os trabalhos citados que estão relacionados à utilização de ontologias para organização dos dados e apresentam maneiras de se executar uma busca conceitual não discutem como tirar proveito dessa classificação na maneira como esses dados serão indexados. A indexação dos dados levando em consideração a similaridade, no nível conceitual, facilitaria a busca por conceitos similares, fazendo com que esse tipo de busca seja menos custosa ao sistema, pois dados considerados conceitualmente similares estariam armazenados próximos uns dos outros no espaço de indexação. Dessa forma, a busca por um grupo de dados se torna mais fácil quando comparada com a busca de um grupo de dados indexados de forma aleatória, por exemplo, utilizando funções de *hash* tradicionais.

Os trabalhos relacionados com o uso de DHTs na indexação de dados que possuam classificação semântica não se preocupam em minimizar o esforço necessário para buscar um grupo de conceitos com certo grau de similaridade entre eles. Geralmente, esses trabalhos indexam cada dado várias vezes, utilizando várias funções de uma família de funções LSH, na tentativa de aumentar a probabilidade de dados similares serem indexados nos mesmos nós, como visto em [44]. Isso pode acarretar em uma grande sobrecarga no sistema do ponto de vista do volume de dados indexados. O trabalho apresentado em [44] indexa textos utilizando funções LSH em suas palavras-chave, indexando cada texto por volta de 20 vezes. Em outras palavras, para cada documento de texto indexado, 20 chaves serão indexadas no sistema. Essa não parece ser uma boa opção à medida que o volume de dados indexados aumenta.

Esses são os pontos principais tratados nesta tese. A proposta apresentada tem como objetivo relacionar dados por seus valores conceituais, combinando a classificação conceitual definida por ontologias e funções LSH, o que permite e facilita a busca conceitual em sistemas distribuídos, como

por exemplo, uma DHT.

As contribuições desta tese podem ser resumidas em:

- apresentar uma metodologia que utiliza funções LSH na criação de identificadores para a indexação de conceitos de ontologias simples (*lightweight*) mantendo a similaridade entre eles, permitindo a busca por similaridade conceitual que, por consequência, facilita a busca por similaridade de conteúdo;
- apresentar resultados que mostram que a metodologia de utilização de função LSH, apresentada nesta tese, agrega os conceitos de uma ontologia *lightweight* em um espaço de indexação, sendo que tal agregação segue a similaridade definida por diferentes métodos de similaridade entre conceitos de uma mesma ontologia;
- apresentar cenários de aplicação para busca conceitual, baseadas em redes P2P sobre DHTs e mostrar como a utilização de funções LSH na indexação de conceitos facilita a busca de dados conceitualmente similares.

No decorrer desta tese será apresentado como utilizar funções LSH para manter a similaridade conceitual dos dados. A ideia principal é que, se em primeiro lugar há uma busca conceitual, uma busca por conteúdo se tornaria mais fácil, já que o escopo dessa se torna mais limitado. A similaridade entre conceitos pode ser usada tanto para complementar resultados quanto para ordená-los, destacando os que possuem maior semelhança com o conceito buscado. Por exemplo, sejam dois conceitos similares *A* e *B*, uma busca por dados classificados no conceito *A* pode gerar uma lista de dados classificados nesse conceito, seguidos de outros classificados no conceito *B*, dadas as suas similaridades com o conceito *A*.

1.3 Organização da tese

Esta tese foi dividida em sete capítulos sendo os quatro primeiros voltados para a fundamentação teórica que será utilizada no restante do trabalho e, a partir do Capítulo 5, é apresentada a proposta de utilização de funções LSH para busca conceitual e cenários de aplicação. Segue uma breve descrição dos capítulos que virão a seguir.

- **Capítulo 2 - Modelagem do conhecimento**: esse capítulo apresenta formas de modelagem de conhecimento, citando diversos tipos de conhecimento utilizados pelos seres humanos e como esses podem ser representados em sistemas computacionais. É apresentado também o conceito de ontologia e como essa pode ser usada como esquema de dados, na Ciência da

Computação, para processamento e inferência de informações. A utilização de domínios de conhecimento definidos utilizando ontologias, empregadas na classificação de dados, é um dos pontos principais desta tese. Nesse capítulo também são apresentadas a Web Semântica e a busca conceitual, contextos em que as propostas desta tese encontram motivação.

- **Capítulo 3 - Similaridade**: apresenta o conceito de similaridade encontrado na literatura, mostrando sua importância para o processamento do conhecimento humano. Nesse capítulo são apresentadas algumas formas de se medir a similaridade entre dois elementos, representados de diversas maneiras como, por exemplo, entre textos, cadeias de caracteres, vetores em um espaço vetorial, etc. Há um maior destaque para as medidas de similaridade entre conceitos de uma mesma ontologia, por ser o foco desta tese.

- **Capítulo 4 - Funções sensíveis à localidade (*Locality Sensitive Hash* - LSH)**: apresenta as funções *hash* sensíveis à localidade (*Locality Sensitive Hash* - LSH). Basicamente, esse tipo de função gera valores de *hash* de maneira que a similaridade entre dois dados, medida por uma função de similaridade, seja mantida em alguma métrica. Será mostrado que cada função LSH é definida para uma determinada função de similaridade e para dados representados de maneiras específicas (por exemplo: vetores, conjuntos de atributos, etc). O capítulo apresenta dois exemplos de função LSH que serão utilizadas nas propostas desta tese.

- **Capítulo 5 - Utilização das funções LSH a partir da classificação conceitual dos dados**: apresenta a proposta de utilização de funções LSH com a classificação conceitual dos dados definida por uma ontologia. A partir de uma classificação semântica, foi utilizada uma função LSH que relaciona dados similares conceitualmente, facilitando uma busca por conceito em um sistema de indexação.

- **Capítulo 6 - Testes e análise**: apresenta cenários de aplicação e testes utilizando a função LSH baseada em ontologia. No capítulo são mostrados dois cenários baseados em redes P2P construídas sobre DHTs. Um desses cenários utiliza a distância Euclidiana como métrica de proximidade e o outro utiliza a distância de Hamming. Em ambos os casos os testes foram bem sucedidos, reduzindo o esforço necessário para recuperar dados similares referentes ao número de saltos na DHT.

- **Capítulo 7 - Conclusão**: apresenta a conclusão da tese e discute os trabalhos futuros.

Capítulo 2

Modelagem de conhecimento

Conhecimento é o entendimento sobre um assunto [50]. Esse entendimento inclui conceitos e fatos sobre esse determinado assunto, assim como as relações entre eles e como essas podem ser utilizadas para gerar novas informações.

A Inteligência Artificial (IA) é o ramo da Ciência da Computação que estuda a natureza do conhecimento humano e como transmitir esse conhecimento para máquinas, de forma que essas possuam ou simulem a capacidade racional e resolvam problemas. A área da IA define alguns conceitos básicos sobre conhecimento como *armazenamento*, *recuperação*, *raciocínio* e *aquisição*, definidos em [2] como a seguir:

- *armazenamento de conhecimento*: é o processo de inserção de conhecimento em um sistema computacional. Esse conhecimento é codificado de maneira que possa ser explorado pelo sistema;
- *recuperação de conhecimento*: é o processo de recuperar uma informação armazenada, de maneira codificada, em um sistema computacional;
- *raciocínio*: é o processo de se utilizar informações de uma base de conhecimento para definir conclusões, inferências e explicações;
- *aquisição de conhecimento*: é o processo de se adquirir, organizar e estruturar conhecimentos sobre um tópico ou domínio, de forma a formatá-los para inserí-los no sistema.

Em IA, todos esses processos são estudados a partir de observações do comportamento humano. Utilizando os princípios citados que os humanos são capazes de, ao se depararem com uma situação nunca vista, conseguirem se adaptar a ela, baseando-se em experiências anteriores. Por exemplo, se um usuário está acostumado a utilizar uma determinada interface gráfica em seu computador, ao ser

apresentado a uma outra, esse terá mais facilidade de compreender a interface nova devido ao fato de já possuir experiência com a interface anterior.

A Ciência Cognitiva identifica vários tipos de conhecimento que os humanos comumente utilizam, os quais são considerados na área de IA. A existência dessa variedade de tipos de conhecimento indica a capacidade humana de organizar, estruturar e utilizar experiências prévias na resolução de problemas. Alguns dos tipos de conhecimento utilizados pela IA, como mostrados em [2], são:

- *conhecimento procedural*: é o tipo de conhecimento relacionado a procedimentos e resolução de problemas. Geralmente esse tipo de conhecimento pode ser definido, por exemplo, em forma de algoritmo;
- *conhecimento declarativo*: descreve o que é conhecido sobre um tópico ou um problema, definindo detalhes de um objeto. Pode apresentar tanto fatos que são verdadeiros quanto fatos falsos;
- *metaconhecimento*: é o conhecimento sobre o conhecimento. É utilizado para determinar se um conhecimento é adequado ou inadequado para tentar resolver um problema;
- *conhecimento heurístico*: inclui regras que guiam o processo de resolução de um problema baseado em experiências anteriores, intuições e habilidades individuais e um bom entendimento do problema;
- *conhecimento estrutural*: descreve estruturas e modelos para organizar problemas e soluções. Define relações entre as diferentes partes do conhecimento de outras categorias. Essas relações são geralmente do tipo “é um” e “parte de”;
- *conhecimento inexato e incerto*: descreve problemas, tópicos e situações nas quais a informação é imprecisa, indisponível, incompleta, aleatória ou ambígua;
- *conhecimento de senso comum*: termo utilizado para descrever o grande volume de conhecimento humano sobre o mundo que não pode ser colocado facilmente em um sistema computacional [51];
- *conhecimento ontológico*: uma representação do conhecimento de um domínio específico. Descreve os conceitos dos elementos daquele domínio e os termos padronizados utilizados pelas pessoas que interagem naquele domínio [52].

A Tabela 2 (adaptada de [50] em [2]) resume esses tipos de conhecimento, apresentando os elementos que os compõem.

Tab. 2.1: Tipos de conhecimento humano [2].

Tipo de conhecimento	Elementos existentes
Procedural	regras, estratégias, agendas, procedimentos
Declarativo	conceitos, objetos, fatos
Metaconhecimento	conhecimento sobre outros tipos de conhecimento
Heurístico	regras gerais
Estrutural	conjuntos de regras, relações entre conceitos e conceitos e objetos
Inexato ou incerto	probabilidades, incertezas, conjuntos e lógica *fuzzy*
Senso comum	proposições padrão, aproximações, hierarquias e analogias gerais
Ontológico	conceitos, relações entre conceitos, axiomas, restrições

Um dos desafios de se utilizar esses diversos tipos de conhecimentos em sistemas computacionais está relacionado com a forma em que esses podem ser representados. A representação deve ser feita de tal maneira que o conteúdo principal do conhecimento seja aproveitado de forma correta pelo sistema. A próxima seção apresenta algumas das formas de representação de conhecimento em um sistema computacional.

2.1 Representação do conhecimento

Em sistemas computacionais, a melhor forma de se representar um conhecimento depende de cada aplicação. Diferentes cenários de aplicação demandam diferentes formas de representação de conhecimento. Todas as representações são imperfeitas e qualquer imperfeição pode originar erros, mas por outro lado, uma representação adequada de um conhecimento é o meio para uma computação eficiente sobre aquele conhecimento [53].

Na literatura são encontradas várias formas de representação do conhecimento em sistemas computacionais que descrevem os tipos de conhecimento descritos na seção anterior. A seguir, como discutido em [2], serão mostradas algumas dessas formas de representação:

- *triplas do tipo objeto-atributo-valor*: esse tipo de representação, também conhecida como triplas OAV, representa objetos e seus atributos, definindo valores para esses atributos. Por exemplo a frase “A camiseta é vermelha” pode ser escrita como um tripla OAV “Camiseta-cor-vermelha”, como visto na Figura 2.1.

  Cada objeto pode possuir vários atributos, permitindo a criação de várias triplas para descrevê-lo;

- *fatos incertos*: uma forma de estender o conceito de uma tripla OAV é inserir um fator de

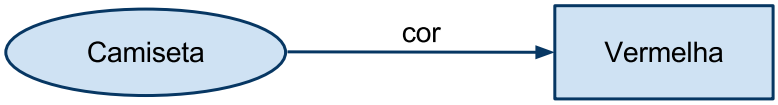


Fig. 2.1: Representação de uma tripla OAV.

certeza (ou incerteza) em sua descrição. Por exemplo, é possível descrever a certeza de um fato definindo as seguintes faixas de valores: *falso* = -1, *provavelmente falso* = -0,5, *desconhecido* = 0, *provavelmente verdadeiro* = 0,5, *verdadeiro* = 1. Os valores definidos para a quantificação da incerteza da afirmação podem ser granularizados da forma que for necessária, para uma maior precisão na tripla. Dessa maneira, um fato incerto, como a frase "A camiseta provavelmente é cara" pode ser escrita como uma tripla OAV com fator de certeza "Camiseta-preço-caro, 0,7", como visto na Figura 2.2.

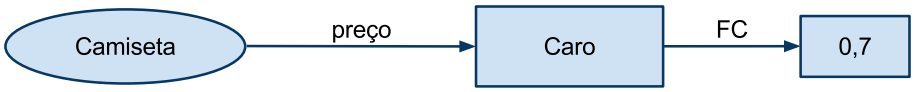


Fig. 2.2: Representação de uma tripla OAV com fator de certeza FC.

- *fatos fuzzy*: representam incertezas usando termos imprecisos e ambíguos comumente utilizados em linguagem natural. Por exemplo, a frase "A camiseta é velha" pode ser interpretada de várias formas, dependendo do conceito de "velha" de quem recebe a informação.

- *regras*: é uma técnica que relaciona premissas (condições ou antecedentes), ou uma situação, a uma ou mais conclusões (consequências) ou ações. Exemplo:

  SE - Está chovendo
  E - É preciso sair à rua
  ENTÃO - É preciso usar um guarda-chuva.

  Geralmente as premissas são representadas por fatos e as conclusões ou ações podem gerar um novo fato ou executar uma ação, invocando um procedimento. É possível gerar regras com fator de incerteza e regras do tipo *fuzzy*;

- *redes semânticas*: também chamadas de mapas de conceitos, são representações que tentam refletir a cognição, ou seja, a maneira como as informações são relacionadas pelos seres humanos [54]. As redes semânticas representam objetos, conceitos e situações de um determinado domínio de conhecimento, conectados por relações. A Figura 2.3 mostra essa representação apresentando os conceitos na forma de elipses e os atributos como quadrados, sendo que um

conceito pode ter diversos atributos. A partir da representação mostrada na Figura 2.3, é possível inferir vários fatos: *Futebol* é um *Esporte* do tipo *Coletivo*, assim como *Basquete* e *Vôlei*. No *Futebol* participam 11 jogadores por equipe enquanto que no *Basquete* e *Vôlei* participam 5 e 6, respectivamente. *Natação* e *Judô* também são do tipo *Esporte*, mas de natureza *Individual*.

Não há um padrão para as redes semânticas serem representadas graficamente, mas tipicamente são representados conceitos (classes), suas instâncias (objetos), atributos desses conceitos, relações entre classes e objetos e valores de atributos.

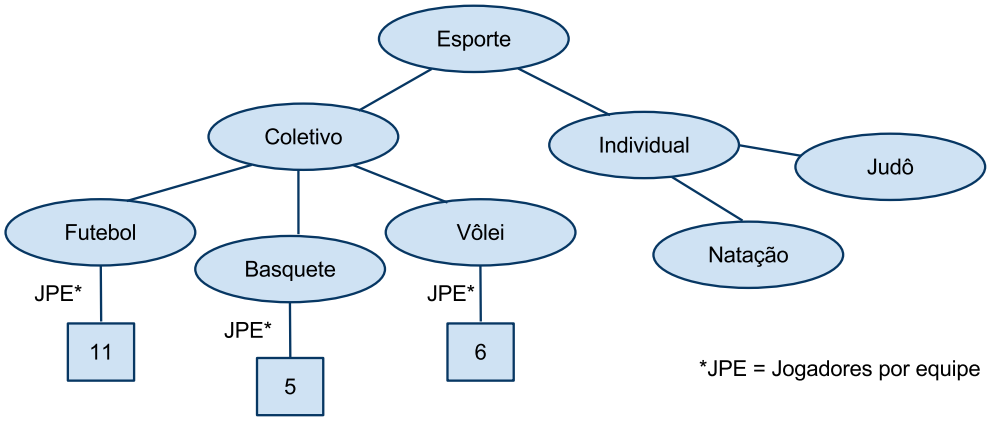


Fig. 2.3: Exemplo de representação de um domínio em rede semântica.

As relações podem ser de qualquer tipo, porém as mais comuns são “é um” e “é parte de”. Uma vantagem das redes semânticas é a facilidade de inserção de novos conceitos e relações. Uma desvantagem é a falta de expressividade para a representação de exceções;

- *frames*: são similares a classes e objetos na programação orientada a objetos. Existem as “classes *frames*” que representam as características comuns de um conjunto de objetos similares, por exemplo *Pessoa*, *Carro*, etc. “*Frames* de instância” representam instâncias específicas de uma classe, por exemplo *Márcia* pode ser uma instância de *Pessoa*. Cada *frame* possui *slots* que representam seus atributos. *Frames* e redes semânticas são muito similares, porém a primeira possui algumas técnicas não possíveis de se executar nas redes semânticas, como as “facetas”. Nos *frames*, as facetas são atreladas a *slots* e possuem procedimentos que são invocados se o valor do atributo representado pelo *slot* for alterado.

Como visto, são vários os tipos de conhecimentos processados pelo ser humano e várias são as representações possíveis para esses conhecimentos em um sistema computacional. Uma das formas mais importantes de se definir um tipo de conhecimento, geralmente utilizada para definir um domínio, é pelo emprego de ontologias. A próxima seção apresentará as ontologias, mostrando a

representação dessas como redes semânticas, discutidas anteriormente. Esse tipo de conhecimento e representação serão importantes para o decorrer desta tese.

2.2 Ontologias

A palavra ontologia vem do grego *ontos* que significa "ser" e *logia* que significa "conhecimento". Na Filosofia remete ao estudo do ser, mais precisamente, ao estudo de categorias das coisas que existem ou podem existir em algum domínio. Uma ontologia de domínio define e explica as coisas existentes naquele domínio [2].

Em [55], o autor define ontologia como uma "especificação de uma conceituação". Por conceituação entende-se o ato de descrever e definir um domínio de conhecimento, por meio da definição de seu vocabulário, seus conceitos e entidades, assim como as relações que podem existir entre si. Especificação significa uma representação formal e descritiva de algo.

Em Ciência da Computação, ontologia pode ser vista como um modelo de dados que representa um conjunto de conceitos existentes em um domínio de conhecimento, definindo as relações entre eles e as regras para a realização de inferência e raciocínio sobre dados. A especificação de um determinado domínio de conhecimento traz benefícios quando o conhecimento é compartilhado, estabelecendo um padrão que deve ser seguido, evitando ambiguidades e interpretações equivocadas. Esse fato é especialmente importante quando há a necessidade de integração de dados provenientes de fontes heterogêneas, como em pesquisas feitas em conjunto por vários grupos de cientistas e pesquisadores de diversas áreas [56].

Em [16], o autor cita as seguintes áreas nas quais as ontologias são especialmente úteis:

- colaboração: ontologias provêem o compartilhamento de conhecimento entre membros de times interdisciplinares e em comunicação entre agentes;
- interoperação: ontologias facilitam a integração de informações, especialmente em aplicações distribuídas;
- educação: ontologias podem ser usadas como meio de publicação e fonte de referência;
- modelagem: ontologias representam blocos reutilizáveis em sistemas de modelagem no nível do conhecimento.

Uma característica das ontologias que pode dificultar seu uso é o fato delas poderem ser tão complexas quanto se queira. Uma forma de se definir ontologias mais simples, diminuindo sua complexidade, é a utilização de ontologias *lightweight* (também chamadas de taxonomias ou hierarquias

de conceitos ISA) [57]. Essa estrutura é modelada como uma hierarquia de conceitos de um determinado domínio, estabelecendo as relações entre eles. Uma forma comum de se representar ontologias *lightweight* é utilizar uma rede semântica, apresentada na seção anterior. A Figura 2.3 representa, em uma rede semântica, uma ontologia *lightweight* para o domínio *Esporte*.

Resumindo, pode-se afirmar que a partir de qualquer ontologia é possível definir uma ontologia *lightweight*, sendo essa mais simples, porém, menos expressiva [2]. Geralmente uma ontologia *lightweight* possui somente relações do tipo "é um" ou "é parte de" entre seus conceitos. Esse é o tipo de ontologia que será utilizada no decorrer dessa tese, inclusive nos cenários de aplicação.

Esse modelo de dados é muito importante no cenário atual da Web Semântica, que será apresentado na próxima seção. A definição de domínios e classificação de dados em modelos como os aqui apresentados é determinante para que o valor semântico dos dados seja bem explorado pelos sistemas, permitindo inferências entre si.

2.3 Web Semântica

A *World Wide Web* (WWW) foi criada por Tim Berners-Lee na tentativa de resolver problemas relacionados ao gerenciamento de um grande volume de dados disponíveis no CERN[1] [58]. A WWW foi concebida como uma infraestrutura descentralizada baseada na interligação de documentos por *hyperlinks*. Navegando pelos *hyperlinks*, o usuário pode "saltar" de um documento a outro, sem precisar se preocupar em quais servidores esses estão localizados. Inicialmente, essa abordagem foi feita para se navegar por documentos dentro do CERN, mas com o uso de protocolos padrões na Internet, como a pilha TCP/IP, o emprego da WWW se alastrou interligando documentos armazenados em diferentes servidores, de diferentes organizações, em diferentes lugares do globo. Com isso, barreiras técnicas e físicas foram quebradas [59].

Do ponto de vista das máquinas, a WWW, como originalmente concebida, é uma rede de documentos interligados, sem informação alguma em relação a essa interligação. Em 2001, Tim Berners-Lee definiu a ideia da Web Semântica [12], uma evolução da WWW que tem o intuito de complementá-la. De maneira simplificada, pode-se afirmar que a Web Semântica é uma camada sobre a WWW convencional, na qual dados são classificados pelo que significam e de maneira que possam ser entendidos pelas máquinas, propiciando a inferência e o raciocínio sobre eles, abrindo novas possibilidades de aplicações antes impossíveis de serem realizadas.

Na WWW convencional, um usuário pode facilmente identificar o significado dos documentos e das interligações entre eles. Por exemplo, ao navegar pelo *website* de uma empresa, o usuário facil-

[1] A sigla CERN vem do francês *Conseil Européen pour la Recherche Nucléaire*. Em 1954 esse conselho foi dissolvido dando origem à uma nova organização, que recebeu o nome de *European Organization for Nuclear Research*, entretanto, a sigla CERN foi mantida.

mente reconhece que aquela página descreve uma empresa, localizada em certo endereço, que atua em determinados ramos e oferece certos serviços. Ao acessar a seção do *website* que lista os seus funcionários, possivelmente existindo *hyperlinks* que interligam aquela seção às páginas pessoais dos funcionários, o usuário facilmente compreende a relação implícita daquela ligação entre os diferentes documentos. A Web Semântica tem como intuito transferir esse conhecimento para as máquinas. Como um exemplo simples, o *website* de uma empresa pode ser classificado como o conceito *Empresa*, dentre os conceitos que definem um domínio de conhecimento. Esse conceito pode ter relações do tipo "trabalha" em ou "emprega" com um conceito do tipo *Pessoa* e assim por diante. Nas palavras de Tim Berners-Lee, enquanto a WWW convencional é uma "rede de documentos", a Web Semântica é uma "rede de coisas", pois nela é possível a existência de uma representação virtual para qualquer entidade e relação do mundo real.

O Consórcio World Wide Web (*World Wide Web Consortium* - W3C)[2] define padrões para a Web Semântica, como o *Resource Description Framework* (RDF) [60] e a *Web Ontology Language* (OWL) [61]. Esses padrões são utilizados para descrever a classificação e atributos dos dados, no caso do RDF, e descrever domínios de conhecimentos, ou ontologias, no caso do OWL. A Figura 2.4 apresenta as camadas padronizadas da Web Semântica[3].

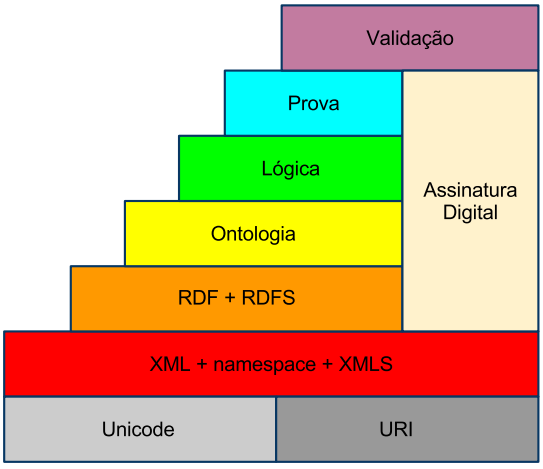


Fig. 2.4: Camadas da Web Semântica.

As camadas apresentadas na Figura 2.4 podem ser descritas como a seguir [62]:

- camada Unicode e URI (*Uniform Resource Identifier*): essa camada define como os dados serão codificados, do ponto de vista da representação dos caracteres e como serão identificados. O padrão Unicode define a codificação dos caracteres e a URI identifica um dado na Internet;

[2]Comunidade internacional que desenvolve padrões com o objetivo de garantir o crescimento da web (http://www.w3.org/ - acessado em 09 de abril de 2011)

[3]Na literatura são encontradas outras versões na organização das camadas da Web Semântica. É possível dizer que a Web Semântica não possui um modelo único de camadas e encontra-se em processo de evolução

- camada XML/XMLS (*Extensible Markup Language* e *XML Schema*) e *namespace*: define que os dados sejam estruturados utilizando XML/XMLS e sigam um espaço de nomes (*namespace*) bem definido;
- camada RDF/RDFS (*Resource Description* e *RDF Schema*): define o modelo básico para a representação de fatos, no caso o RDF, e uma estrutura conceitual simples, no caso o RDFS;
- camada de ontologia: utiliza linguagens mais expressivas que o RDFS. Atualmente, na Web Semântica é comumente utilizada a linguagem OWL (*Web Ontology Language*);
- camada lógica: é utilizada para aumentar ainda mais a expressividade da camada de ontologia e permitir a determinação de conhecimentos declarativos específicos para determinadas aplicações;
- camada de prova: envolve o processo de dedução e comprovação das informações das camadas inferiores;
- camada de validação: envolve a validação das assinaturas digitais das informações providas pelas camadas inferiores. Trata-se de uma camada importante, pois é essencial que os usuários confiem nos serviços oferecidos pela Web Semântica, para que essa possa ser usada em sua totalidade.

Aplicações que desejam explorar essas informações atreladas aos dados, ou seja, esse "novo" tipo de dado, devem ser capazes de executar as seguintes tarefas, citadas em [63]:

- encontrar fontes de dados relevantes: é desejado que as aplicações possuam maneiras de encontrar fontes de dados relevantes de maneira dinâmica, durante seu funcionamento;
- selecionar conhecimento apropriado: as aplicações devem selecionar os conhecimentos e dados apropriados dentre aqueles que foram previamente analisados, a partir de critérios previamente determinados e específicos, para aquele tipo de aplicação;
- explorar fontes heterogêneas de dados: as aplicações devem usar dinamicamente dados heterogêneos e genéricos;
- combinar ontologias e recursos: devido à grande variedade de fontes e tipos de dados, as aplicações devem combinar conhecimentos de diversas fontes e utilizá-los.

Como visto, quase todos os itens citados acima remetem à busca e processamento de dados semânticos, desde encontrar fontes de dados relevantes, selecionar o conhecimento apropriado até combinar

recursos. Para lidar com este paradigma, no qual os dados possuem muito mais valor semântico, novos métodos de busca são necessários. Em [64] é feita uma revisão de vários métodos de busca semântica, classificando-os como métodos orientados a documentos, orientados a entidades e conhecimento, para busca de conteúdo multimídia, centrados em relações, baseados em análise semântica e baseados em mineração de dados. Esses tipos são explicados a seguir.

- **Métodos orientados a documentos**: o principal objetivo é a busca por documentos tais como páginas *web*, publicações científicas ou mesmo ontologias. Nesta categoria, os documentos possuem anotações semânticas, geralmente em RDF, seguindo uma ontologia de domínio, provendo informação sobre os tópicos e o conteúdo dos documentos. O processo de busca é feito comparando-se termos pesquisados pelo usuário com os existentes nas anotações semânticas;

- **métodos orientados a entidades e conhecimentos**: esses métodos expandem o escopo das buscas convencionais, que passa a não buscar somente documentos. É utilizada a relação entre termos de um mesmo domínio, definidos pelas ontologias, para a exploração e navegação através dos dados. Resultados mais amplos são providos, mostrando relações e atributos das entidades recuperadas;

- **métodos para busca de conteúdo multimídia**: nesta categoria, conteúdos multimídia (imagem, vídeo, áudio) possuem anotações semânticas baseadas em ontologias. Esse método é similar ao método orientado a documentos, mas existem alguns desafios adicionais devido ao fato que a extração de informações semânticas de um documento, por exemplo suas palavras-chave, é mais simples do que a extração dessas informações em conteúdos multimídia. O processo de busca é o mesmo, buscando por anotações dos conteúdos;

- **métodos centrados em relações**: esses métodos usam termos implícitos utilizados pelos usuários. Analisadores léxicos ou processadores de inferência são aplicados utilizando uma base de conhecimento. Sinônimos são empregados para aumentar o escopo da busca, assim como a combinação de sujeito, predicado e objeto;

- **métodos baseados em análise semântica**: nesta categoria, soluções baseadas na teoria dos grafos são usadas para representar, interpretar e descobrir relações complexas entre os recursos. Bases RDF são modeladas como grafos diretos e as arestas são utilizadas como caminhos entre um elemento e outro;

- **métodos baseados em mineração de dados**: a maior parte das informações que podem ser recuperadas não está indexada, não possui anotações semânticas ou não estão relacionadas com outras informações. Dessa forma, alguns tipos de informações são mineradas de grandes

quantidades de dados. Por exemplo, a questão "quais foram os produtos agrícolas mais produzidos no Brasil no ano de 2010?" somente pode ser respondida após a análise de uma grande quantidade de dados (documentos, páginas *web*, etc), usando alguma técnica de inferência de conhecimento por mineração de dados.

Esta tese tem como foco a busca por entidades e conhecimento, na qual o alvo da busca não é um documento específico de um certo tipo, mas dados de vários tipos que compartilham a mesma classificação ontológica dada por uma ontologia de domínio. A partir da definição de um domínio de conhecimento, é possível descrevê-lo em conceitos e classificar dados. Seguindo a proposta da Web Semântica, podemos citar, como exemplo, vários projetos que utilizam classificação semântica por hierarquia de conceitos para diversos domínios diferentes. Por exemplo, *MusicMoz*[4], *Musicbrainz*[5], *Yahoo!*[6] são alguns exemplos de base de dados que possuem uma classificação conceitual para definir o domínio *Música*. Esse tipo de classificação possibilita a busca conceitual, na qual um usuário não busca por um dado específico, mas sim dados referentes a um mesmo conceito. Por exemplo, com a estrutura lógica da Web Semântica, é possível buscar "todos os funcionários da empresa X". A próxima seção apresenta definições para os conceitos relacionados à busca conceitual.

2.4 Busca Conceitual

Vários trabalhos na literatura consideram somente o conteúdo dos dados para criar relações entre eles. Imagens são relacionadas entre si pelas suas características visuais (histograma, formas geométricas) [24][23], textos são relacionados pelas palavras-chave que possuem [20][65], etc. Acima da similaridade de conteúdo há um outro nível de similaridade, no qual dados de vários tipos (imagens, vídeos, textos, etc) podem ser agrupados pela sua classificação em um domínio de conhecimento, geralmente definido por um conjunto de conceitos. Por exemplo, um texto sobre futebol, um vídeo sobre um jogo de futebol e uma foto de um momento importante de um jogo de futebol podem ser agrupados em um mesmo conceito, por exemplo *Futebol*. Utilizando essa abordagem, é possível executar a *busca conceitual* [13][14][15], na qual o objetivo é recuperar dados classificados pelo mesmo conceito em um domínio de conhecimento. Esse tipo de busca é mais ampla que as buscas baseadas em palavras-chave, como dito em [13]. Em uma busca baseada em palavras-chave, é necessária a existência dessas palavras-chave, ou sinônimos, nos dados buscados, por exemplo: palavras em um texto, marcações nas anotações de uma imagem, etc. Na busca conceitual isso não é necessário pois, não importando seu conteúdo, os dados podem ser classificados em um mesmo conceito específico

[4] *MusicMoz* utiliza uma hierarquia de conceitos - http://musicmoz.org (acessado em 30 de março de 2011).
[5] *Musicbrainz* utiliza tags conceituais relacionadas - http://musicbrainz.org (acessado em 30 de março de 2011).
[6] Diretórios de categorias do *Yahoo!* - http://dir.yahoo.com (acessado em 30 de março de 2011).

do domínio. Em uma busca por um conceito, todos os dados classificados naquele conceito fazem parte da resposta.

O uso de hierarquias de conceitos, também chamadas de ontologias *lightweight* ou grafos conceituais, pode ser encontrado em vários trabalhos na literatura. Grafos conceituais são utilizados em vários tipos trabalhos, como, por exemplo, na busca de textos referentes a leis [35], informações médicas [34] e documentos multimídia [36].

No domínio legal, referente a leis e assuntos relacionados com advocacia, a busca conceitual é muito importante, como visto em [35]. Por meio dela é possível a busca de documentos relacionados a um conceito, sem a necessidade de se buscar um caso específico. É possível recuperar informações sobre os conceitos envolvidos em um caso, mesmo que sejam usadas variações dos termos específicos contidos nos documentos, provendo ao usuário qualquer informação armazenada na base de dados sobre os conceitos relacionados.

No caso de documentos médicos, como afirmado em [34], as técnicas de recuperação de informação e descoberta de conhecimento focam no processamento de dados altamente estruturados, em particular dados organizados em um formato de atributo-valor. Para dados não estruturados, a abordagem convencional é o processamento por palavras-chave existentes nos dados indexados. Essa abordagem impossibilita buscas mais complexas do ponto de vista clínico e epidemológico que requerem, por exemplo, diagnósticos, tratamentos e resultados dos exames feitos, relacionados a cada caso. Os autores desse trabalho examinaram e comprovaram a eficiência da busca conceitual para resolver esse tipo de problema. A busca foi feita baseada em grafos conceituais utilizados como formalismo para a representação de conhecimento médico contido em documentos.

Em se tratando de busca por imagens, [36] afirma que é preciso formas de explicitar a relação semântica entre as palavras-chave que descrevem dados. Dessa forma os resultados das buscas se tornam melhores se comparados com aqueles obtidos somente por análise de palavras-chave. Essa necessidade se torna ainda maior quando os dados são imagens, uma vez que essas podem ser ainda melhor descritas por meio de conceitos.

Utilizando a busca conceitual, apoiada em domínios de conhecimento definidos por hierarquias de conceitos, o usuário pode procurar informações sobre um ou mais conceitos que estejam indexadas em uma base de conhecimento, mesmo que essas informações não sejam comparáveis entre si, isto é, sejam de diferentes tipos. A Figura 2.5 mostra como dados de diversos tipos podem ser comparados conceitualmente.

Na Figura 2.5, os conjuntos A e B podem representar imagens de determinadas raças de cães, enquanto o conjunto C representa textos sobre adestramento canino e os três conjuntos, apesar da diferença de conteúdo, podem estar relacionados com o mesmo conceito D, por exemplo, *Cães*. Em uma busca por similaridade de conteúdo, comumente encontrada na literatura, não seria possível, por

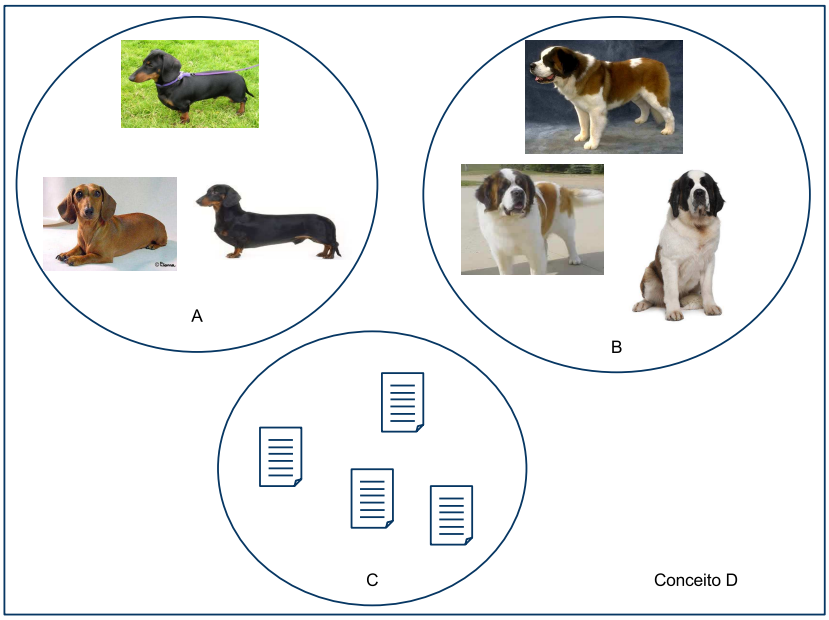


Fig. 2.5: Exemplo de relação conceitual entre dados de diferentes tipos. A e B são conjuntos de imagens e C é um conjunto de textos.

exemplo, a partir de uma imagem de um cão da raça Dachshund encontrar imagens referentes a outras raças de cães, ou mesmo textos sobre o assunto. É desse desafio que a busca conceitual trata.

Em um cenário como o da Web Semântica, apresentado na seção anterior, no qual os dados possuem um valor conceitual definido por uma ontologia simples associada a um domínio de conhecimento, a busca conceitual se torna naturalmente viável e de muita importância no tratamento de grandes volumes de dados e descoberta de informação.

2.5 Sumário

O ser humano utiliza diferentes tipos de conhecimento para resolver problemas. Para que esses conhecimentos sejam utilizados em sistemas computacionais, é preciso representá-los de forma a permitir uma computação eficiente das informações que expressam. Existem diversas maneiras de se representar conhecimentos em sistemas computacionais, cada uma com suas características e níveis de expressividade. Uma forma de definir um domínio de conhecimento é por meio de ontologias, muito importantes para o cenário atual da Web Semântica. Ontologias que, por sua vez, podem ser representadas, por exemplo, por meio de redes semânticas. A representação de uma ontologia deve ser capaz de permitir a extração de informações e regras estabelecidas, assim como, a inferência de novos conhecimentos que não estejam representados de forma explícita.

Este capítulo apresentou algumas formas de representação de conhecimento em sistemas compu-

tacionais. O próximo capítulo apresenta o conceito de similaridade encontrado na literatura, o qual é fundamental para os cenários de busca conceitual tratados nesta tese.

Capítulo 3

Similaridade

O conceito de similaridade é explorado na Psicologia há muitos anos. Amos Tversky, um dos pioneiros da Ciência Cognitiva[1], afirma em [3] que a similaridade possui um papel fundamental nas teorias do conhecimento e comportamento, servindo como um princípio organizacional pelo qual os indivíduos classificam objetos, formam conceitos e fazem generalizações. Em [66][67][68] é afirmado que em teorias de categorização assume-se que a classificação de novos exemplos é feita baseada na sua similaridade com exemplos ou categorias já conhecidos.

A literatura possui vários trabalhos desenvolvidos na tentativa de criar métricas para determinar a similaridade entre conceitos de uma mesma hierarquia conceitual. O estudo da similaridade semântica entre conceitos tem feito parte do processamento de linguagem natural e recuperação de informação por vários anos [69]. A maneira como os humanos processam informações geralmente envolve comparação de conceitos, existindo várias maneiras de se medir a similaridade entre conceitos, dependendo da representação adotada para o conhecimento [70]. Alguns exemplos de utilização de similaridade entre conceitos incluem a eliminação de ambiguidade do significado dos conceitos, detecção de erros em um texto, recuperação de imagens, entre outros. O problema da avaliação da relação semântica entre conceitos em uma representação por rede semântica possui uma longa história nas áreas de IA e Psicologia e similaridade semântica é um caso especial de relacionamento semântico. Por exemplo, como visto em [71], os conceitos *carro* e *gasolina* podem ser mais fortemente relacionados do que *carro* e *bicicleta*, mas o segundo par certamente é mais similar.

O valor de uma medida de similaridade está em sua utilidade para um determinado objetivo [71]. No domínio cognitivo, a similaridade é tratada como uma propriedade caracterizada pela percepção e intuição humanas, da mesma maneira como plausibilidade e tipicalidade. Dessa forma, o valor de uma medida de similaridade está na sua fidelidade ao comportamento e percepção humanas, como

[1]Ciência Cognitiva é o estudo interdisciplinar da mente e da inteligência. Fazem parte de Ciência Cognitiva a Psicologia, a Filosofia, a Inteligência Artificial, a Neurociência e a Lingüística.

medido em trabalhos experimentais.

Atualmente, com o crescimento da Web Semântica, a classificação conceitual dos dados disponíveis na Web ganha maior importância. Na Web Semântica um texto pode ser classificado pelo seu assunto e não somente pelas palavras-chave e termos que contém, como visto no Capítulo 2, o que facilita a busca por textos diferentes, mas relacionados.

Uma maneira de organizar um domínio de conhecimento no qual dados são classificados é utilizando ontologias simples. Vários são os projetos encontrados na comunidade *Linked Data*[2] que disponibilizam dados classificados por uma ontologia *lightweight*, como definida no Capítulo 2. Essa organização de um domínio por ontologias simples possibilita a busca conceitual por similaridade, uma vez que permite a utilização de métodos para determinar a similaridade entre os diferentes conceitos da ontologia.

Na próxima seção será definido o conceito de funções de similaridade, utilizadas para determinar a similaridade entre elementos comparáveis de um determinado conjunto.

3.1 Definição do conceito de funções de similaridade

Uma função de similaridade tem a finalidade de medir o grau de semelhança entre dois elementos e pode ser definida como:

$$sim(a, b) \rightarrow [0, 1]$$

sendo *a* e *b* objetos de um determinado domínio *U* no qual os elementos são comparáveis em alguma métrica. Quanto maior o valor retornado pela função de similaridade, ou seja, mais próximo do valor 1, mais similares são os objetos. Caso contrário, quanto mais próximos do valor 0, os objetos são considerados menos similares. Como visto em [72], uma função de similaridade implementa um algoritmo para definir a similaridade no intervalo [0, 1]. Esse valor pode tanto ser utilizado para determinar se *a* é similar ou não a *b*, como para estabelecer uma relação de relevância entre um determinado elemento e os outros presentes no conjunto universo. Por exemplo, dado um objeto *a*, esse pode ser comparado com todos os outros pertencentes ao conjunto *U*, utilizando-se a função de similaridade. O valor obtido pela função pode ser utilizado para ordenar os outros elementos, formando assim uma lista (*ranking*) em relação à relevância dos elementos em relação ao objeto *a*.

A representação em sistemas computacionais dos diferentes tipos de dados existentes pode ser feita de várias formas, o que implica na necessidade de vários tipos de medidas de similaridade. A próxima subseção define algumas dessas formas de se medir similaridade para diferentes representações dos dados.

[2] *Linked Data* - http://linkeddata.org/ (acessado em 18 de Março de 2011).

3.1.1 Medidas de similaridade

Em [73], o autor divide as medidas de similaridade em cinco categorias:

- similaridade entre vetores;
- similaridade entre conjuntos;
- similaridade em textos (incluindo similaridade em cadeias de caracteres);
- similaridade em teoria da informação;
- similaridade em grafos (incluindo medidas em árvores).

Segue uma breve descrição de cada uma delas.

Similaridade entre vetores

Nesse tipo de abordagem, os dados são representados por vetores que podem ser comparados entre si. Vetores que representam dados similares possuirão características similares entre si.

Uma medida de similaridade comumente usada é o cálculo do cosseno do ângulo entre dois vetores. Quanto mais similares, menor o ângulo entre eles, e por consequência, o cosseno desse ângulo possui valor mais próximo de 1. A Figura 3.1 mostra um exemplo para vetores normalizados[3].

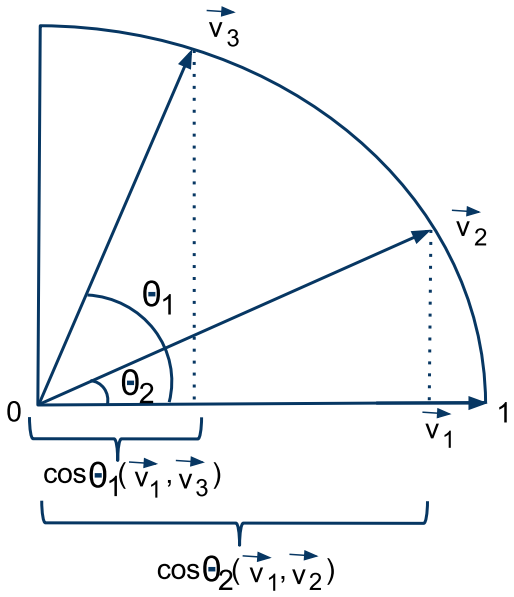


Fig. 3.1: Representação do cosseno do ângulo entre vetores.

[3]Vetores normalizados possuem tamanho igual a 1.

Para cada par de vetores podemos calcular o cosseno do ângulo entre eles da seguinte forma:

$$\cos\theta(\overrightarrow{u}, \overrightarrow{v}) = \frac{|\overrightarrow{u}.\overrightarrow{v}|}{|\overrightarrow{u}||\overrightarrow{v}|} \tag{3.1}$$

na qual $\overrightarrow{u}$ e $\overrightarrow{v}$ são vetores, $|\overrightarrow{u}.\overrightarrow{v}|$ é o produto escalar entre eles e $|\overrightarrow{u}|$ é o tamanho do vetor $\overrightarrow{u}$.

Por exemplo, para três vetores normalizados $\in R^3$, $\overrightarrow{a}$ = (0,577, 0,577, 0,577), $\overrightarrow{b}$ = (0,267, 0,534, 0,802) e $\overrightarrow{c}$ = (0,894, 0, 0,447), a similaridade entre eles pode ser calculada, como a seguir:

$$sim_{cos}(\overrightarrow{a}, \overrightarrow{b}) = \frac{|\overrightarrow{a}.\overrightarrow{b}|}{|\overrightarrow{a}||\overrightarrow{b}|} \approx 0,924$$
$$sim_{cos}(\overrightarrow{a}, \overrightarrow{c}) = \frac{|\overrightarrow{a}.\overrightarrow{c}|}{|\overrightarrow{a}||\overrightarrow{c}|} \approx 0,772$$
$$sim_{cos}(\overrightarrow{b}, \overrightarrow{c}) = \frac{|\overrightarrow{b}.\overrightarrow{c}|}{|\overrightarrow{b}||\overrightarrow{c}|} \approx 0,596.$$

Isso mostra que os vetores $\overrightarrow{a}$ e $\overrightarrow{b}$ são mais similares que o par $\overrightarrow{a}$ e $\overrightarrow{c}$ e $\overrightarrow{c}$ é mais similar a $\overrightarrow{a}$ do que a $\overrightarrow{b}$.

Para vetores binários, cujas coordenadas assumem os valores 0 ou 1, uma outra medida de similaridade possível é a similaridade de Hamming, definida como:

$$sim_{Hamming}(\overrightarrow{a}, \overrightarrow{b}) = \frac{\textit{número de bits coincidentes}}{\textit{número total de bits dos vetores}}$$

no qual $\overrightarrow{a}$ e $\overrightarrow{b}$ são vetores binários de mesmo tamanho. Essa medida de similaridade está diretamente relacionada com a distância de Hamming, sendo calculada pela contagem do número de bits não coincidentes entre duas cadeias binárias. O número de bits diferentes constitui na distância de Hamming entre uma cadeia e outra.

Por exemplo, as sequências binárias 00010110 e 00001110 possuem os seguintes valores para similaridade e distância de Hamming:

$$sim_{Hamming} = \tfrac{6}{8} = 0,75$$

$$dist_{Hamming} = 2.$$

Similaridade entre conjuntos

Uma forma de se representar dados é por um conjunto de características ou atributos. Dois dados similares possuirão conjuntos com elementos em comum, existindo um grande número de elementos na intersecção entre seus conjuntos. Uma medida comum de similaridade entre conjuntos é o índice de Jaccard, definido como:

$$sim_{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

sendo *A* e *B* conjuntos, $|A \cap B|$ o número de elementos contidos na intersecção entre os conjuntos *A* e *B* e $|A \cup B|$ o número de elementos contidos na união dos dois conjuntos.

Dessa forma, quanto maior a intersecção entre dois conjuntos, mais similares eles são considerados. Como mostrado na Figura 3.2, os conjuntos A e B possuem maior intersecção, ou seja, maior quantidade de elementos em comum, B e C possuem poucos elementos em comum e A e C não possuem intersecção alguma. Dessa forma, segundo o índice de Jaccard, A e B possuem maior similaridade entre si do que B e C, que são pouco similares. Os conjuntos A e C não possuem similaridade alguma.

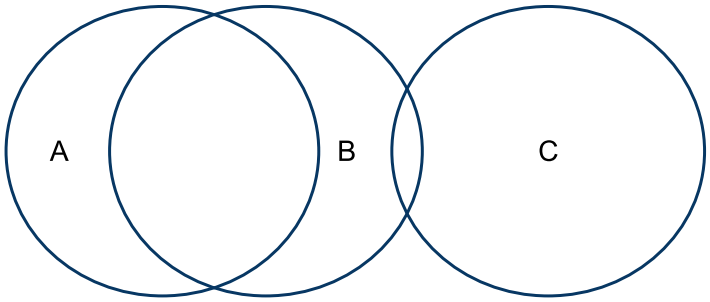


Fig. 3.2: Representação da intersecção de três conjuntos.

Em um exemplo prático, sejam *D*, *E* e *F* conjuntos de inteiros *D* = {0, 1, 2, 3, 4, 5, 6, 7, 8}, *E* = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} e *F* = {2, 3, 4, 8, 9, 10}. O índice de Jaccard para esses conjuntos é igual a:

$$sim_{Jaccard}(D, E) = \frac{9}{10} = 0{,}9$$
$$sim_{Jaccard}(D, F) = \frac{4}{11} = 0{,}363$$
$$sim_{Jaccard}(E, F) = \frac{5}{11} = 0{,}454.$$

Similaridade em textos

Nesse tipo de similaridade, textos e palavras são comparados entre si. Textos podem ser representados como um conjunto de palavras-chave, o que pode ser resumido em um problema de comparação entre conjuntos, apresentado anteriormente. É possível também comparar cadeias de caracteres, encontrando aquelas mais similares umas às outras. A medida de similaridade para cadeia de caracteres mais comum é a distância de Levenshtein [74]. Basicamente, essa medida de distância contabiliza o número de inserções, remoções e operações de substituição que devem ser feitas em uma cadeia de caracteres para que ela se torne idêntica à outra. Quanto menos operações devem ser feitas, maior a similaridade entre elas.

Por exemplo, a distância Levenshtein entre as palavras “computador” e “computar” é 3, pois são necessárias 3 edições para transformar a primeira na segunda:

computador → computa**r**or (substituição da letra d pela r);
computaror → computa**r**r (remoção da letra o);
computarr → **computar** (remoção da letra r).

Similaridade em teoria da informação

Esse tipo de abordagem se apóia na noção de conteúdo da informação para calcular a similaridade entre entidades (por exemplo ontologias, conceitos em uma taxonomia ou códigos fonte). O termo conteúdo da informação é originado em [75] e pode ser descrito como a quantidade de informação necessária para descrever ou contida em uma entidade. Algumas abordagens, por exemplo, contabilizam que quanto mais específica uma entidade, em outras palavras, quanto mais baixa na hierarquia que define um domínio, mais informação essa entidade possui [76].

Similaridade em grafos e árvores

Similaridades em grafos e árvores são especialmente interessantes para a Web Semântica e busca conceitual devido ao fato de ontologias e hierarquias de conceitos poderem ser utilizadas para definir domínios, sendo representadas nas linguagens RDF/OWL. Dessa forma, soluções para similaridade nesses modelos podem ser utilizadas no contexto da Web Semântica.

Os diversos métodos existentes para calcular esse tipo de similaridade leva em consideração características topológicas das estruturas, como a distância em número de vértices entre dois conceitos, a profundidade do conceito em comum entre dois outros, etc. Mais detalhes serão dados na Seção 3.2.

3.1.2 Revisão do uso de similaridade

Como visto no Capítulo 1, vários trabalhos encontrados na literatura utilizam similaridade para relacionar diversos tipos de dados. Textos [44], imagens [23][24], vídeos [27] e áudio [26] são exemplos de tipos de dados relacionados por similaridade em diversos trabalhos. Uma característica desses trabalhos é a relação entre os dados do ponto de vista do seu conteúdo. Esse fato impossibilita a comparação de dados que possuam conteúdos e tipos diferentes, mas que podem ser conceitualmente iguais como, por exemplo, um texto e uma foto sobre um mesmo assunto.

A próxima seção apresenta como é possível relacionar por similaridade conceitos de uma mesma hierarquia. Esse tipo de relacionamento entre conceitos permite que dados diferentes, mas classificados em um mesmo conceito, ou conceitos similares, sejam relacionados entre si, facilitando a busca conceitual, como visto na Seção 2.4.

3.2 Similaridade entre conceitos de uma ontologia

Como já dito anteriormente, existem métodos para contabilizar a similaridade entre elementos de uma mesma hierarquia de conceitos. Dessa forma, dada uma hierarquia, que pode ser uma ontologia *lightweight*, é possível identificar quais os conceitos que são mais similares entre si. Vários parâmetros podem ser levados em consideração nesse cálculo, como será visto nas próximas subseções.

A seguir serão apresentados alguns métodos de similaridade entre conceitos de uma mesma ontologia. Como apresentado em [10], os métodos podem ser baseados em três aspectos: (i) em relação a características topológicas da ontologia, tais como o menor caminho entre os conceitos e a profundidade do conceito mais geral em comum entre dois conceitos; (ii) o conteúdo da informação da ontologia; e (iii) um glossário de termos da ontologia. Esta tese discute apenas métodos de similaridade baseados no aspecto topológico da ontologia, por entender que os outros aspectos necessitam da utilização de diferentes técnicas que abrangem outros escopos de pesquisa, como a comparação entre ontologias do ponto de vista de vocabulários que não sigam uma padronização, análise léxica, etc. Esses outros escopos fogem daquele apresentado aqui, apesar de não serem concorrentes e sim complementares. As análises da similaridade definida pelos métodos serão feitas em relação à ontologia apresentada na Figura 3.3.

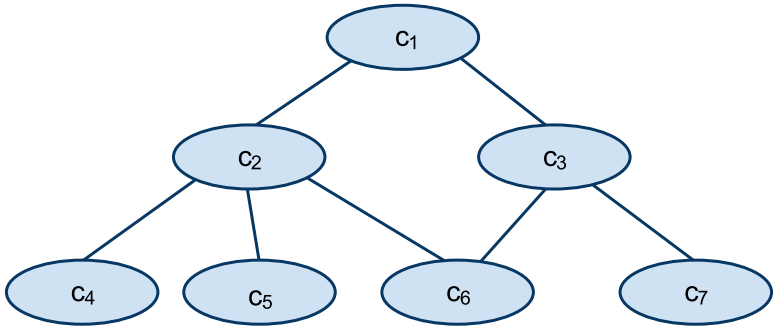


Fig. 3.3: Exemplo de ontologia *lightweight* [1].

3.2.1 Menor caminho

Como apresentado em [77], os autores visam resolver um problema como o definido a seguir: imagine um usuário navegando pela estrutura de um *website* e esse esteja interessado em encontrar mais informações sobre um determinado assunto. O usuário, então, pergunta: "Encontre-me mais documentos relacionados e similares ao tópico que é classificado na ontologia utilizada neste *site* como *CONCEITO X*". Esse tipo de busca é um exemplo claro de busca conceitual, como definido no Capítulo 2. Em [77] é introduzida uma distância ontológica entre conceitos, definida como $Ds(c_i, c_j)$,

como sendo o menor caminho entre os conceitos c_i e c_j, se esse existir e 0 caso contrário. Essa é uma métrica simples, também utilizada em [70], que somente considera a distância entre os conceitos de uma ontologia, sem levar em conta o nível de generalização ou especialização dos conceitos. Dois conceitos que estão próximos são considerados similares sem considerar a profundidade desses na ontologia. A Tabela 3.2.1 apresenta as similaridades para os conceitos da ontologia da Figura 3.3. Para se obter uma métrica no intervalo [0, 1], na qual quanto mais próximo de 0, menos similar e quanto mais próximo de 1, mais similar, os valores foram invertidos, ou seja, para cada par de conceitos foi contabilizado 1 - $Ds(c_i, c_j)$. Todos os valores foram normalizados pelo valor do maior entre todos os menores caminhos encontrados. A Figura 3.4 ilustra a similaridade definida por esse método em relação a um conceito específico, neste caso c_7; quanto mais escura a cor de um conceito na figura, mais similar esse é em relação a c_7. No decorrer desta tese esse método será referenciado como *Bouquet* em homenagem ao autor principal do artigo referenciado. O mesmo padrão será adotado nos próximos métodos apresentados.

Tab. 3.1: Tabela de similaridade do método *Bouquet*.

Conceito	c_1	c_2	c_3	c_4	c_5	c_6	c_7
c_1	1	0,75	0,75	0,5	0,5	0,5	0,5
c_2	0,75	1	0,5	0,75	0,75	0,75	0,25
c_3	0,75	0,5	1	0,25	0,25	0,75	0,75
c_4	0,5	0,75	0,25	1	0,5	0,5	0
c_5	0,5	0,75	0,25	0,5	1	0,5	0
c_6	0,5	0,75	0,75	0,5	0,5	1	0,5
c_7	0,5	0,25	0,75	0	0	0,5	1

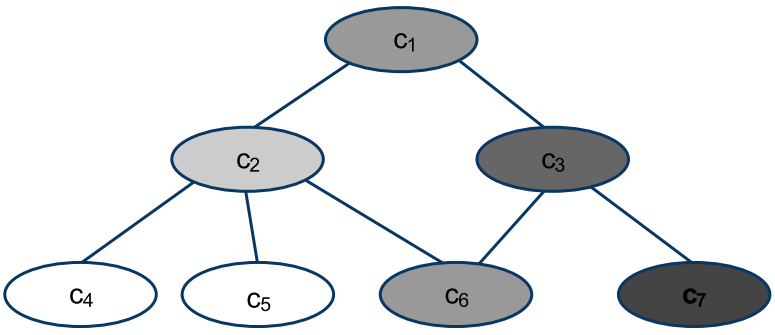


Fig. 3.4: Representação visual da similaridade definida pelo método *Bouquet* em relação ao conceito c_7. Quanto mais escura a cor de um conceito, mais similar a c_7.

3.2.2 Menor caminho escalado

Os autores de [78] atacam o problema da identificação do significado de uma palavra. Em um experimento, eles tentam explorar medidas de similaridade baseadas na *WordNet*[4]. O propósito é mostrar como palavras semanticamente similares provêem sugestões contextuais similares. Por exemplo, se, na língua inglesa, a palavra *baseball* define bem um dos sentidos da palavra *play*[5], portanto outras palavras similares a *baseball*, tais como *football* ou *basketball*, devem definir o mesmo significado de *play*. Seguindo essa abordagem, é possível notar que cada uma dessas palavras também pode ser um conceito em uma hierarquia de conceitos ou ontologia simples.

A medida de similaridade apresentada em [78] é definida como:

$$sim(c_i, c_j) = max\left[-log\left(\frac{length(c_i, c_j)}{2D}\right)\right]$$

na qual $length(c_i, c_j)$ é o tamanho de um caminho entre os conceitos c_i e c_j. O menor caminho entre os conceitos c_i e c_j é aquele que gera o valor máximo obtido pela equação. Na mesma equação, D é a profundidade máxima da ontologia. Essa métrica é, basicamente, a métrica do menor caminho escalada pela profundidade da ontologia e com uma distribuição logarítmica. A Tabela 3.2.2 apresenta as similaridades para os conceitos da ontologia da Figura 3.3, para esse método de similaridade. A Figura 3.5 ilustra a similaridade definida por esse método em relação a um conceito específico, neste caso c_7; quanto mais escura a cor de um conceito na figura, mais similar este é em relação a c_7. No restante desta tese esse método será chamado de *Leacock*.

Tab. 3.2: Tabela de similaridade do método *Leacock*.

Conceito	c_1	c_2	c_3	c_4	c_5	c_6	c_7
c_1	1	0,6	0,6	0,3	0,3	0,3	0,3
c_2	0,6	1	0,3	0,6	0,6	0,6	0,12
c_3	0,6	0,3	1	0,12	0,12	0,6	0,6
c_4	0,3	0,6	0,12	1	0,3	0,3	0
c_5	0,3	0,6	0,12	0,3	1	0,3	0
c_6	0,3	0,6	0,6	0,3	0,3	1	0,3
c_7	0,3	0,12	0,6	0	0	0,3	1

[4]Base de dados lexical da língua inglesa, organizada como uma hierarquia de conceitos (http://wordnet.princeton.edu - acessado em 30 de março de 2011).

[5]O verbo, em inglês, *to play* possui vários significados como brincar, jogar, tocar (um instrumento musical), entre outros.

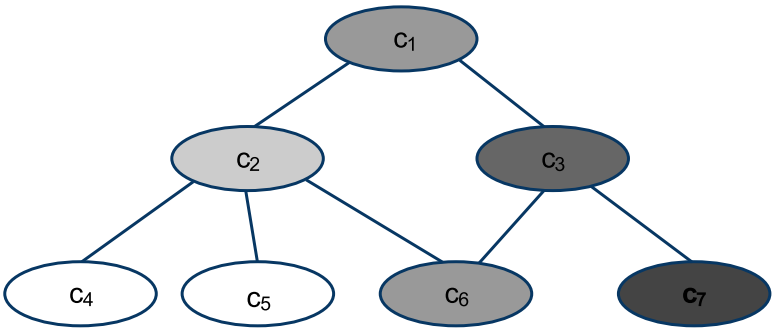


Fig. 3.5: Representação visual da similaridade definida pelo método *Leacock* em relação ao conceito c_7. Quanto mais escura a cor de um conceito, mais similar a c_7.

3.2.3 Profundidade da ontologia

Em [79], os autores abordam o problema da tradução automática entre línguas. O estudo mostra que, na tradução de verbos do inglês para o chinês e vice-versa, é difícil obter uma boa cobertura dos significados apenas listando pares traduzidos. Os autores propõem uma representação semântica para os verbos, definindo cada verbo por um conjunto de conceitos em diferentes domínios conceituais. Baseada nessa representação conceitual, uma medida de similaridade pode ser definida. Essa abordagem pode também ser empregada em hierarquias conceituais ou ontologias simples. Em [79] é apresentada uma métrica baseada no comprimento do caminho entre os conceitos e a raiz da hierarquia, ou seja, a profundidade dos conceitos. O método é redefinido em [71] com a seguinte equação:

$$sim(c_i, c_j) = \frac{2.depth(lcs(c_i,c_j))}{depth(c_i)+depth(c_j)}$$

na qual $lcs(c_i, c_j)$ é o conceito mais profundo em comum entre os conceitos c_i e c_j, em outras palavras, o primeiro conceito em comum nos caminhos de c_i e c_j até a raiz e $depth(c_n)$ é a distância do conceito c_n até a raiz. Essa métrica encontra a profundidade do lcs dos conceitos e escala o resultado pela soma da profundidade dos próprios conceitos. É interessante notar que, se lcs é também a raiz da hierarquia, a similaridade entre ambos os conceitos é 0. Seguindo a lógica da métrica, se dois conceitos pertencem a dois diferentes ramos da ontologia, não existindo nenhum ancestral especializado em comum entre eles, a não ser o mais genérico deles (a raiz), significa que eles não possuem nenhuma relação entre si. A similaridade de dois conceitos aumenta quando o lcs entre eles é mais profundo na ontologia e esses não se encontram muito distantes, em termos de profundidade, do lcs. Na Tabela 3.2.3 são mostradas as similaridades dos conceitos da Figura 3.3. A Figura 3.6 ilustra a similaridade definida por esse método em relação a um conceito específico, neste caso c_7; quanto mais escura a cor de um conceito na figura, mais similar esse é em relação a c_7. No restante deste texto esse método será chamado de *Wu*.

Tab. 3.3: Tabela de similaridade do método *Wu*.

Conceito	c_1	c_2	c_3	c_4	c_5	c_6	c_7
c_1	1	0	0	0	0	0	0
c_2	0	1	0	0,67	0,67	0,67	0
c_3	0	0	1	0	0	0,67	0,67
c_4	0	0,67	0	1	0,5	0,5	0
c_5	0	0,67	0	0,5	1	0,5	0
c_6	0	0,67	0,67	0,5	0,5	1	0,5
c_7	0	0	0,67	0	0	0,5	1

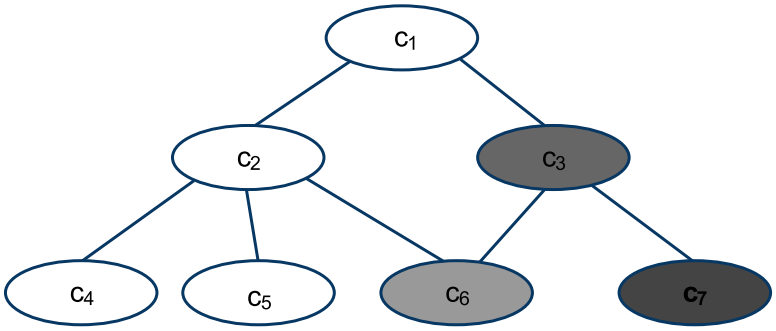


Fig. 3.6: Representação visual da similaridade definida pelo método *Wu* em relação ao conceito c_7. Quanto mais escura a cor de um conceito, mais similar a c_7.

3.2.4 Combinação entre menor caminho e profundidade

A similaridade entre palavras em uma hierarquia é abordada em [69]. Os autores definem que, para encontrar a similaridade semântica entre duas palavras, é necessário considerar o tamanho do caminho entre elas, a profundidade dos conceitos e a densidade local. O trabalho formula várias estratégias, combinando esses atributos. A estratégia que apresenta o melhor resultado combina o menor caminho e a profundidade dos conceitos, definida pela seguinte equação:

$$sim(c_i, c_j) = \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & \text{se } c_i \neq c_j \\ \\ 1 & \text{caso contrário.} \end{cases}$$

Na equação, l é o tamanho do menor caminho entre os conceitos c_i e c_j, h é a profundidade do conceito direto em comum entre c_i e c_j e $\alpha \geq 0$ e $\beta \geq 0$ são parâmetros que escalam a contribuição de l e h, respectivamente[6].

A equação para dois conceitos diferentes é composta por duas partes. A primeira parte, $e^{-\alpha l}$, aumenta a similaridade quando l diminui e a segunda parte, $\frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}}$, aumenta a similaridade quando

[6]Em [69] é indicado que os melhores valores para α e β são 0,2 e 0,6, respectivamente.

o valor de h aumenta. Em outras palavras, a similaridade entre os conceitos aumenta quando a profundidade do lcs aumenta, ou seja, ambos conceitos se tornam mais especializados e o caminho entre eles diminui. Na Tabela 3.2.4 são encontradas as similaridades definidas por esse método para os conceitos da Figura 3.3. A Figura 3.7 ilustra a similaridade definida por esse método em relação a um conceito específico, neste caso c_7; quanto mais escura a cor de um conceito na figura, mais similar esse é em relação a c_7. A partir deste ponto, esse método será referenciado como *Li*.

Tab. 3.4: Tabela de similaridade do método *Li*.

Conceito	c_1	c_2	c_3	c_4	c_5	c_6	c_7
c_1	1	0	0	0	0	0	0
c_2	0	1	0	0,44	0,44	0,44	0
c_3	0	0	1	0	0	0,44	0,44
c_4	0	0,44	0	1	0,36	0,36	0
c_5	0	0,44	0	0,36	1	0,36	0
c_6	0	0,44	0,44	0,36	0,36	1	0,36
c_7	0	0	0,44	0	0	0,36	1

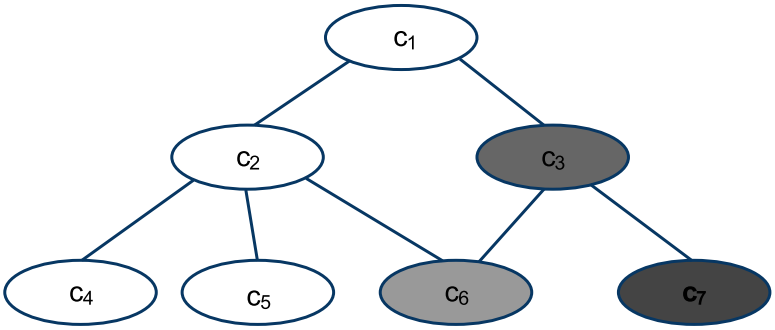


Fig. 3.7: Representação visual da similaridade definida pelo método *Li* em relação ao conceito c_7. Quanto mais escura a cor de um conceito, mais similar a c_7.

3.2.5 Similaridade pela representação dos conceitos em um espaço vetorial

A técnica "Modelo avançado de espaço vetorial baseado em tópicos" (*Enhanced Topic-based Vector Space Model* (eTVSM)), avaliada em [1], apresenta um exemplo de representação de conceitos de uma ontologia *lightweight* em um espaço vetorial. Essa técnica constrói um mapa de tópicos para a comparação conceitual entre palavras de um texto que consiste em um grafo direto com tópicos representados como nós e as arestas representando as relações entre eles. Esse tipo de mapa de tópicos é similar a uma ontologia *lightweight*.

Um dos passos utilizados pelo eTVSM é criar vetores conceituais para cada conceito da ontologia, dessa forma é possível obter a similaridade entre esses conceitos. A similaridade entre dois conceitos é definida pelo cosseno do ângulo entre os pares de vetores, como visto na Seção 3.1.1. Para cada conceito é possível obter o seu respectivo vetor por meio do algoritmo apresentado a seguir.

Para todos os conceitos folha c_i de uma ontologia de *n* conceitos, cada vetor de conceito $\overrightarrow{c_i}$ = {$c_{i,1}$, $c_{i,2}$, ..., $c_{i,n}$} é calculado da seguinte forma:

$$c_{i,j} = \begin{cases} 1 & \text{se } c_j \in \text{caminho de } c_i \text{ até a raiz ou } i = j \\ 0 & \text{caso contrário.} \end{cases}$$

Para todos os conceitos internos c_i, o vetor de conceito é definido por:

$$\overrightarrow{c_i} = |\sum \overrightarrow{c_s}|$$

no qual $\overrightarrow{c_s}$ são os vetores de conceito de todos os filhos diretos do conceito c_i.

Por exemplo, os vetores de conceitos da ontologia 3.3, após normalização, são os seguintes:

$\overrightarrow{c_1}$ = {0,669, 0,495, 0,429, 0,120, 0,120, 0,255, 0,174}
$\overrightarrow{c_2}$ = {0,642, 0,642, 0,194, 0,224, 0,224, 0,194, 0}
$\overrightarrow{c_3}$ = {0,607, 0,282, 0,607, 0, 0, 0,282, 0,325}
$\overrightarrow{c_4}$ = {0,577, 0,577, 0, 0,577, 0, 0, 0}
$\overrightarrow{c_5}$ = {0,577, 0,577, 0, 0, 0,577, 0, 0}
$\overrightarrow{c_6}$ = {0,5, 0,5, 0,5, 0, 0, 0,5, 0}
$\overrightarrow{c_7}$ = {0,577, 0, 0,577, 0, 0, 0, 0,577}

A similaridade entre os conceitos é dada pelo cosseno do ângulo entre eles, calculado como mostrado na Equação 3.1 (Seção 3.1.1). Quanto maior o ângulo entre os vetores, menor o valor do cosseno desse ângulo e, por consequência, menor a similaridade entre eles. Por outro lado, quanto menor o ângulo entre os vetores, maior o valor do cosseno desse ângulo e, por consequência, maior a similaridade entre eles. A Tabela 3.2.5 mostra os valores do cosseno do ângulo entre os vetores apresentados.

Nesse método, a forma pela qual são criados os vetores conceituais dos conceitos internos, ou seja, somando os vetores que representam os conceitos que são seus filhos diretos, faz com que conceitos sejam considerados mais similares àqueles mais genéricos do que aqueles que são seus vizinhos. Isso se dá pois a soma entre dois vetores resulta em um vetor situado entre ambos, como

Tab. 3.5: Tabela de similaridade do método *eTVSM*.

Conceito	c_1	c_2	c_3	c_4	c_5	c_6	c_7
c_1	1	0,933	0,933	0,741	0,741	0,924	0,734
c_2	0,933	1	0,742	0,871	0,871	0,836	0,483
c_3	0,933	0,742	1	0,513	0,513	0,888	0,888
c_4	0,741	0,871	0,513	1	0,667	0,577	0,333
c_5	0,741	0,871	0,513	0,667	1	0,577	0,333
c_6	0,924	0,836	0,888	0,577	0,577	1	0,577
c_7	0,734	0,483	0,888	0,333	0,333	0,577	1

mostrado na Figura 3.8. É possível ver isso na Tabela 3.2.5. Por exemplo, para o conceito c_4, a lista de similaridades com os outros conceitos da ontologia, em ordem decrescente, é c_2, c_1, c_5, c_6, c_3 e c_7, ou seja, “sobe-se” na hierarquia para depois considerar conceitos mais próximos e profundos.

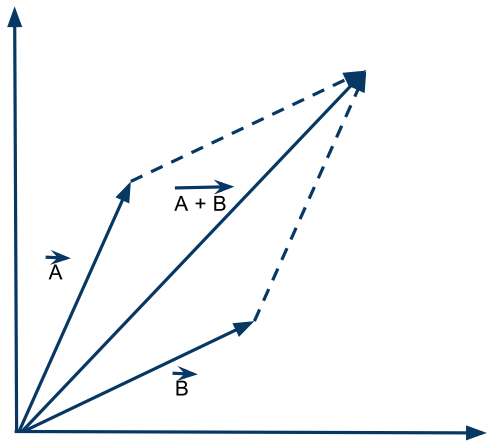


Fig. 3.8: Soma vetorial.

A Figura 3.9 ilustra a similaridade definida por esse método em relação a um conceito específico, neste caso c_7; quanto mais escura a cor de um conceito na figura, mais similar esse é em relação a c_7.

3.2.6 Discussão sobre os diferentes métodos

Como visto anteriormente, existem vários métodos para medir a similaridade entre dois conceitos em uma hierarquia de conceitos, ou uma ontologia simples. Existem também vários parâmetros que podem ser utilizados para se contabilizar essa similaridade como, por exemplo, o menor caminho entre dois conceitos, a altura em que esses conceitos se encontram na hierarquia, a altura do primeiro conceito em comum aos dois conceitos no caminho desses até a raiz, etc.

De acordo com a necessidade do sistema e do tipo de ontologia que será alvo de análise de simi-

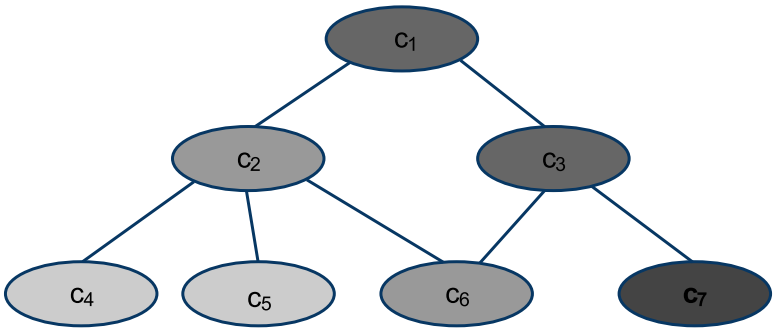


Fig. 3.9: Representação visual da similaridade definida pelo método *eTVSM* em relação ao conceito c_7. Quanto mais escura a cor de um conceito, mais similar a c_7.

laridade, um dos métodos pode ser mais indicado que outro. Uma ontologia pode retratar somente uma especialização de um conceito dentro de um domínio específico como, por exemplo, a ontologia apresentada na Figura 2.3 na qual todos os conceitos derivam de um mesmo conceito dentro do domínio, no caso, *Esporte*. Por outro lado, algumas ontologias podem agrupar domínios diferentes abaixo de um conceito raiz genérico, por exemplo, *Coisa*. Isso faz com que alguns métodos considerem conceitos diretamente ligados ao conceito raiz como similares devido, por exemplo, à proximidade entre eles, mesmo que esses definam domínios diferentes. A ontologia da Dbpedia[7] [80] é um exemplo de ontologia na qual são agregados vários domínios, todos derivando de um mesmo conceito totalmente genérico, no caso, *Thing*. Esse tipo de característica da ontologia deve ser levado em consideração no momento de se escolher qual método utilizar.

A respeito dos métodos apresentados neste capítulo, é possível identificar semelhanças e diferenças em relação à maneira como esses calculam a similaridade entre dois conceitos. Os métodos *Bouquet* e *Leacock* são similares em relação aos seus comportamentos, já que ambos utilizam somente o menor caminho entre os conceitos para definir a similaridade entre eles. Isso faz com que conceitos localizados próximos uns dos outros, na ontologia, sejam considerados similares, não importando a altura desses na estrutura. O método baseado no eTVSM possui um comportamento no qual um conceito é primeiro considerado mais similar aos conceitos mais gerais que o geram, e depois aos conceitos mais especializados e próximos a ele. E mesmo conceitos distantes e de diferentes ramos da ontologia, em relação à raiz, possuem um certo grau de similaridade, mesmo que pequeno. Isso pode resultar em dois conceitos próximos, mas que definam dois ramos de especialização diferentes na ontologia, serem considerados similares. Essa situação ocorre, por exemplo, com os conceitos c_2 e c_3 na ontologia da Figura 3.3. Outra característica desse tipo de método é a geração de valores de similaridades que podem ser melhor aproveitados em ontologias que definam conceitos mutuamente

[7]Dbpedia é um projeto que disponibiliza arquivos RDF que descrevem conceitualmente os artigos encontrados na Wikipedia (http://www.dbpedia.org - acessado em 28 de Abril de 2011).

disjuntos. Pela lógica do método, seguindo a ontologia da Figura 3.3, se c_4 e c_5 são conceitos filhos de c_2, isso quer dizer que c_4 relaciona-se mais com c_2 do que com c_5, mesmo que c_5 tenha a mesma relação de "parentesco" com c_2. Porém, c_4 continua tendo mais relação com c_5 se comparado com um outro conceito que não possua um conceito em comum, no caminho até a raiz, com c_4, por exemplo c_7 que é filho de c_3.

O método *Wu* somente considera a profundidade dos conceitos da ontologia e do seu conceito em comum até a raiz. Isso pode resultar em conceitos, mesmo que distantes na ontologia, mas profundos o suficiente, serem considerados similares. O método *Li* considera tanto o menor caminho quanto a profundidade desses na ontologia para determinar a similaridade entre conceitos. Uma característica desses dois métodos é o fato de considerarem conceitos que estão localizados em diferentes ramos da ontologia, em relação ao conceito raiz, totalmente dissimilares (similaridade igual a 0). Essa característica faz sentido em ontologias que possuem um conceito totalmente genérico como conceito raiz, como a ontologia da *Dbpedia*. Porém, pode não ser interessante quando a raiz é apenas o maior nível de generalização daquele domínio, como no caso da ontologia do domínio *Esporte*.

Outra questão importante é a utilização de escalas para utilizar o valor retornado por um método, sem alterar seu comportamento. Por exemplo, é possível notar que, de acordo com os resultados do método *Li*, mostrados na Tabela 3.2.4, para ontologias com pouca profundidade, o maior nível de similaridade entre conceitos diferentes é aproximadamente 0,44, o que pode ser considerado baixo. Dependendo da maneira que esse método será utilizado, pode ser necessário o uso de uma escala, o que não alteraria em nada o seu comportamento. O mesmo pode ser feito para qualquer método.

Em resumo, é importante compreender os diferentes métodos e suas características no momento de escolher qual utilizar em um sistema computacional. É preciso uma análise sobre qual o tipo de ontologia que será utilizada ou até mesmo adaptá-la para que o método utilizado possa aferir de forma satisfatória o nível de similaridade entre os conceitos.

3.3 Sumário

O estudo do conceito de similaridade é muito importante nas áreas da Psicologia e da Ciência Cognitiva. É por meio da comparação e identificação de características similares entre algo novo e conhecimentos já adquiridos que o ser humano consegue se adaptar a novas situações. A inserção do conceito de similaridade em sistemas computacionais requer a definição de funções que possam medir essa característica entre dados comparáveis, levando em consideração a maneira como esses dados estão representados no sistema.

Este capítulo apresentou o conceito de similaridade encontrado na literatura, mostrando vários tipos de funções de similaridade para diferentes tipos de representação dos dados. Em seguida foi

apresentada a noção de similaridade entre conceitos de uma ontologia, discutindo alguns métodos utilizados para esse cálculo. Existem na literatura vários desses métodos e neste capítulo foram apresentados cinco deles, denominados nesta tese como *Bouquet*, *Leacock*, *Wu*, *Li* e um outro baseado no algoritmo de extração de vetores conceituais utilizado no *eTVSM*. Cada um deles possui diferenças na maneira que relacionam os conceitos, sendo alguns mais simples, como no caso do método *Bouquet*, até alguns mais complexos, que levam em consideração vários aspectos da ontologia, como o método *Li*. Esse tipo de determinação de similaridade será importante para a proposta desta tese, no momento que for combinada com as funções LSH apresentadas no próximo capítulo. Apesar dos vários métodos serem diferentes em sua essência, será mostrado, como uma das contribuições desta tese, que todos podem ser combinados com funções LSH.

Capítulo 4

Funções *hash* sensíveis à localidade (*Locality Sensitive Hash* - LSH)

A quantidade de dados, de vários tipos, disponíveis nos dias de hoje para os usuários da Internet é gigantesca. Por meio de consultas simples nos mais famosos sites de busca, é possível obter milhões de páginas e outros milhões de imagens, vídeos, textos, etc. Um dos desafios orginados por esse cenário atual é a busca por dados similares, seja similaridade conceitual ou de conteúdo.

O crescimento da Web Semântica veio dar suporte a esse tipo de busca e manipulação dos dados, já que essa adiciona aos dados o valor do seu significado conceitual, como visto na Seção 2.3. Por outro lado, pouco foi feito na questão da indexação e armazenamento desses dados, de maneira a facilitar a busca por dados similares.

Este capítulo apresenta as funções *hash* sensíveis à localidade (*Locality Sensitive Hashing* - LSH), as quais foram projetadas para criar identificadores para dados de forma que seja mantida a similaridade entre eles, medida por uma função de similaridade. As funções LSH são tipicamente utilizadas em buscas por dados similares, nas quais o objetivo é encontrar todos aqueles, existentes em uma base de dados, que compartilham um grau de semelhança ao dado utilizado na busca. Sendo a indexação de dados similares feita utilizando o conceito das funções LSH, uma busca por dados que possuam alguma relação entre si é facilitada, demandando menos esforço em um sistema de busca.

Nas próximas subseções serão apresentados os conceitos e exemplos de funções LSH.

4.1 Definição das funções LSH

Em [81] é definido que uma família de funções *hash F* é classificada como sensível à localidade (LSH) correspondente a uma função de similaridade *sim(a, b)*, se, para toda função $h \in F$, temos:

$$\mathbf{Pr}_{h \in F}[h(a) = h(b)] = sim(a, b) \tag{4.1}$$

onde **Pr** é a probabilidade e *sim(a, b)* é uma função de similaridade que varia entre 0 e 1, sendo 1 para *a* e *b* completamente similares e 0 caso contrário.

Em [21] é discutida a existência de famílias de funções *hash* para medidas de similaridade apresentando três lemas, reproduzidos a seguir.

Lema 1 - Para qualquer função de similaridade sim(a, b) que admite uma família de funções LSH segundo a Equação 4.1, a função de distância d = 1 - sim(a, b) satisfaz a desigualdade do triângulo.

Prova - Suponha que exista uma função LSH tal que

$$\mathbf{Pr}_{h \in F}[h(a) = h(b)] = sim(a, b).$$

Então

$$\mathbf{Pr}_{h \in F}[h(a) \neq h(b)] = 1 - sim(a, b).$$

Seja $\delta_h(a, b)$ uma variável que assume valores no conjunto {0, 1} indicando se $h(a) \neq h(b)$; é afirmado que $\delta_h(a, b)$ satisfaz a desigualdade do triângulo pois:

$$\delta_h(a, b) + \delta_h(b, c) \geq \delta_h(a, c). \tag{4.2}$$

Pelos valores assumidos por $\delta_h()$, o único caso que poderia violar a inequação seria se $\delta_h(a, b) = \delta_h(b, c) = 0$ e $\delta_h(a, c) = 1$. Mas nesse caso $h(a) = h(b)$ e $h(b) = h(c)$. Portanto, $h(a) = h(c)$, o que implica que $\delta_h(a, c) = 0$ e a inequação é satisfeita. Isso prova a suposição.

Dessa forma, pelo valor esperado[1] **E** da inequação 4.2 sobre $h \in F$, deduz-se a seguinte equação:

$$\mathbf{E}_{h \in F}[\delta_h(a, b)] + \mathbf{E}_{h \in F}[\delta_h(b, c)] \geq \mathbf{E}_{h \in F}[\delta_h(a, c)].$$

Como $\delta(a, b)$ satisfaz a desigualdade do triângulo, $\mathbf{E}_{h \in F}[\delta_h(a, b)]$ também a satisfaz. Isso prova o lema.

[1]Em Probabilidade, o valor esperado (ou esperança) de uma variável aleatória é a soma das probabilidades de cada possibilidade de saída da experiência multiplicada pelo seu valor. Isto é, representa o valor médio "esperado" de uma experiência se ela for repetida muitas vezes. O valor em si pode não ser obtido, pois esse pode ser improvável ou impossível. Se todos os eventos tiverem igual probabilidade o valor esperado é a média aritmética das probabilidades.

Lema 2 - Dada uma família F de funções LSH correspondente a uma função de similaridade sim(a, b), podemos obter uma família F' de funções LSH que mapeia objetos em {0, 1} e corresponde à função de similaridade $\frac{1+sim(a,b)}{2}$.

Prova - Suponha uma família *F* de funções LSH que siga a equação

$$\mathbf{Pr}_{h\in F}[h(a) = h(b)] = sim(a, b).$$

Seja *B* uma família de funções *hash* independentes que operam no domínio da família *F* e mapeiam os resultados no domínio {0, 1}. Então, se $u \neq v$, a probabilidade é definida como

$$\mathbf{Pr}_{b\in B}[b(u) = b(v)] = \frac{1}{2}$$

e se $u = v$

$$\mathbf{Pr}_{b\in B}[b(u) = b(v)] = 1.$$

Considere a família de funções *hash* obtida ao se aplicar uma função da família *B* no resultado de uma função da família *F*. Essa função mapeia objetos em {0, 1}, sendo

$$\mathbf{Pr}_{h\in F, b\in B}[b(h(u)) = b(h(v))] = \frac{1+sim(u,v)}{2}.$$

Com probabilidade $sim(u, v)$, $h(u) = h(v)$ e, portanto, $b(h(u)) = b(h(v))$. Com probabilidade $1 - sim(u, v)$, $h(u) \neq h(v)$ e, nesse caso

$$\mathbf{Pr}_{b\in B}[b(u) = b(v)] = \frac{1}{2}.$$

Portanto,

$$\mathbf{Pr}[b(h(u)) = b(h(v))] = sim(u, v) + (1 - sim(u, v))\frac{1}{2} = \frac{1+sim(u,v)}{2}.$$

Lema 3 - Para qualquer função de similaridade $sim(a, b)$ *que admite uma função LSH como definido na Equação 4.1, a função de distância* $1 - sim(a, b)$ *é incorporada isometricamente em um cubo de Hamming*[2].

Prova - Aplicando o Lema 2, temos uma família de funções LSH *F* correspondente à função de similaridade $sim'(a, b) = \frac{1+sim(a,b)}{2}$. É possível notar que essa família de funções LSH binária incorpora os objetos no espaço de Hamming pela concatenação dos resultados de todas as funções da família. Para dois objetos x e y, temos que $f(x)$ é o ponto no cubo de Hamming em que x é mapeado, sendo $f(x) = (h_1(x), ..., h_d(x))$ e $d = |F|$. A equação $1 - sim'(a, b)$ é simplesmente a fração dos bits que não coincidem entre $f(x)$ e $f(y)$, ou seja, a distância de Hamming entre $f(x)$ e $f(y)$. Portanto, esse mapeamento é isométrico pela função de distância $1 - sim'(a, b)$ no cubo de Hamming. É possível deduzir que

[2][82] define que um conjunto $A \subset Y$ em um espaço (Y, d) é incorporado isometricamente em um espaço (Y_0, d_0) se existe um mapeamento α de A para Y_0 tal que para qualquer $x, y \in A$ a distância $d(x, y) = d_0(\alpha(x), \alpha(y))$.

$$1 - sim'(a,b) = 1 - \frac{1+sim(a,b)}{2} = \frac{1-sim(a,b)}{2}.$$

Isso implica que $1 - sim(a,b)$ pode ser incorporado isometricamente no cubo de Hamming.

A partir do Lema 3 é possível afirmar que para uma função de similaridade $sim(a,b)$, qualquer incorporação isométrica da distância $1 - sim(a,b)$ no cubo de Hamming gera uma família de funções LSH correspondente à função de similaridade $\frac{\alpha+sim(a,b)}{\alpha+1}$ para $\alpha > 0$.

Os lemas apresentados, como visto em [21], definem características para a existência de funções LSH. Na próxima seção serão apresentados alguns exemplos de funções LSH encontradas na literatura.

4.2 Exemplos de funções LSH

Existem vários tipos de funções LSH, sempre definidas de acordo com algum tipo de representação dos dados e atreladas a uma função de similaridade que compara esses dados. Por exemplo, dados podem ser representados por vetores, que representam suas características, ou por conjuntos de atributos. No primeiro caso, uma forma de se relacionar os dados é calculando o cosseno do ângulo entre os vetores que os representam, enquanto que no segundo, dois conjuntos podem ser relacionados pelo coeficiente de Jaccard. Ambos os casos foram apresentados na Seção 3.1.1.

Nas próximas subseções serão apresentadas duas funções LSH que serão utilizadas no decorrer desta tese. A primeira foi desenvolvida para dados representados como conjuntos e a segunda para dados representados por vetores.

4.2.1 Permutações independentes *Min-wise*

As permutações independentes *Min-wise* (*Min-wise independent permutations*) [18][19][20] provêem uma família de funções LSH para conjuntos de inteiros, sendo a similaridade correspondente ao coeficiente de Jaccard (Capítulo 3). Esse coeficiente é definido como:

$$sim_{Jaccard}(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

onde A e B são conjuntos de inteiros. O coeficiente de Jaccard define que, quanto maior a quantidade de elementos na interesecção de dois conjuntos, mais similares eles são.

Basicamente, a função *hash Min-Wise* define uma permutação aleatória a ser executada em todos os elementos de um conjunto e o menor valor resultante será o valor do *hash*. Quanto maior a intersecção de dois conjuntos A e B, maior a chance de que um dos elementos da intersecção gere o menor valor, fazendo com que os dois conjuntos tenham o mesmo valor de *hash*. Quanto menor a intersecção, menor a chance desse fato ocorrer.

Formalmente, a função é definida da seguinte forma: seja π uma permutação aleatória no universo de inteiros *I*, *A* = {a_1, a_2, ..., a_n} $\subseteq$ *I*, e *B* = {b_1, b_2, ..., b_n} $\subseteq$ *I*, a função de *hash* h_π é definida como:

$$h_\pi(A) = min\{\pi(a_1), \pi(a_2), ..., \pi(a_n)\}$$

onde h_π(*A*) aplica a permutação π em cada elemento de *A* e considera somente o menor dos resultados. Para dois conjuntos *A* e *B*, temos que *x* = h_π(*A*) = h_π(*B*) somente se π^{-1}*(x)* $\in$ (*A* $\cap$ *B*). Portanto, para dois conjuntos *A* e *B*:

$$\mathbf{Pr}_{h\in F}[h(A) = h(B)] = sim(A, B) = \frac{|A\cap B|}{|A\cup B|}.$$

Em [19] é mostrada uma forma de se implementar essa função, descrita a seguir.

Para conjuntos de inteiros de *n* bits cria-se uma chave de *n* bits que tenha exatos $\frac{n}{2}$ bits aleatórios iguais a 1. Para um inteiro *a* de um conjunto, move-se todo bit de *a* que corresponda às posições iguais a 1 da chave para a metade inicial de um novo inteiro e os outros para a metade final, gerando nesse caso *a'*. Em seguida, cria-se uma chave de $\frac{n}{2}$ bits, com exatos $\frac{n}{4}$ bits aleatórios iguais a 1. A chave criada é repetida *x* vezes e concatenada, gerando uma chave de *n* bits (nesse passo *x* é igual a 2) e o mesmo processo anterior é feito sobre *a'* gerando *a"*. Os passos são repetidos até que a chave do *hash* tenha tamanho 2 e cada par de bits do inteiro tenha sido permutado. As mesmas chaves são usadas em todos os elementos do conjunto *A* e de todos os conjuntos que se queira criar valores de *hash*. A Tabela 4.2.1 mostra um exemplo para *n* igual a 8 bits.

Todos os elementos de um conjunto passam pelo mesmo processo e o menor valor gerado será escolhido como o valor do *hash* que representa aquele conjunto.

4.2.2 Funções Hash *Random Hyperplane*

Outra função de similaridade que pode ser empregada é o cosseno do ângulo entre dois vetores que representam dados, como visto na Seção 3.1.1. Sendo θ o ângulo, em radianos, entre os vetores $\overrightarrow{u}$ e $\overrightarrow{v}$, quanto mais próximo de 1 o valor de $\cos\theta(\overrightarrow{u}, \overrightarrow{v})$, mais similares os vetores são. Caso contrário, quanto mais próximo de 0, menor é a relação entre eles. Para $\cos\theta(\overrightarrow{u}, \overrightarrow{v})$ exatamente igual a 1, temos que ambos os vetores são totalmente relacionados e, para esse valor igual a 0, não existe relação alguma entre eles. A equação do cosseno do ângulo entre dois vetores foi definida na Equação 3.1 (Seção 3.1.1).

Uma família de funções LSH que correspondem à similaridade de cosseno é a função *Random Hyperplane Hash* (RHH) [21]. Essa função se baseia no princípio das distribuições "*p*-estáveis" (*p-stable distributions*) [83].

Em [84], as distribuições estáveis são definidas como: uma distribuição *D* sobre o conjunto *R* é chamada p-estável se existe um índice de estabilidade $p \geq 0$ tal que para quaisquer números reais {v_1,

Tab. 4.1: Exemplo de permutação feita pela função LSH *Min-Wise* para um inteiro de 8 bits. Inteiro"' é o resultado final.

Posição do bit	1	2	3	4	5	6	7	8
Chave de 8 bits	1	1	1	0	0	1	0	0
Inteiro	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**1**
Inteiro'	1	0	0	1	1	0	1	1
Posição original do bit	1	2	3	6	4	5	7	8
Posição do bit	1	2	3	4	5	6	7	8
Chave de 4 bits	1	0	0	1	1	0	0	1
Inteiro'	1	0	0	1	1	0	1	1
Inteiro"	1	1	1	1	0	0	0	1
Posição original do bit	1	4	5	8	2	3	6	7
Posição do bit	1	2	3	4	5	6	7	8
Chave de 2 bits	0	1	0	1	0	1	0	1
Inteiro"	1	1	1	1	0	0	0	1
Inteiro"'	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**0**
Posição original do bit	2	1	4	3	6	5	8	7

..., $v_n\}$ e variáveis aleatórias X_1, ..., X_n independentes e identicamente distribuídas com distribuição *D*, a variável aleatória $\sum_i v_i X_i$ possui a mesma distribuição da variável $(\sum_i |v_i|^p)^{\frac{1}{p}} X$ onde *X* é uma variável aleatória com distribuição *D*.

Distribuições estáveis existem para qualquer $p \in (0, 2]$ [85], sendo o exemplo mais conhecido a distribuição Gaussiana (ou Normal), que é 2-estável. A ideia é criar um vetor aleatório que siga uma distribuição estável e utilizá-lo na função de *hash*, calculando o produto vetorial entre o vetor criado e o vetor que representa um dado.

A função RHH é definida como: dada uma coleção de vetores em R^d, um vetor aleatório $\overrightarrow{r}$ é escolhido a partir de uma distribuição Gaussiana *d*-dimensional, ou seja, cada coordenada do vetor $\overrightarrow{r}$ é sorteada de uma distribuição Gaussiana com média igual a 0 e variância igual a 1 (também conhecida como uma distribuição Normal N(0, 1)). Para um vetor $\overrightarrow{r}$ criado dessa forma, a função de *hash* é definida como:

$$h_{\overrightarrow{r}}(\overrightarrow{u}) = \begin{cases} 1 & \text{se } \overrightarrow{r}.\overrightarrow{u} \geq 0 \\ 0 & \text{se } \overrightarrow{r}.\overrightarrow{u} < 0 \end{cases} \tag{4.3}$$

Para vetores $\overrightarrow{u}$ e $\overrightarrow{v}$, [21] mostra que

$$\mathbf{Pr}[h_r(\overrightarrow{u}) = h_r(\overrightarrow{v})] = 1 - \frac{\theta(\overrightarrow{u}, \overrightarrow{v})}{\pi} \approx \cos\theta(\overrightarrow{u}, \overrightarrow{v}) \tag{4.4}$$

pelo fato que $\cos\theta(\overrightarrow{u}, \overrightarrow{v})$ pode ser estimado a partir do valor de $\theta(\overrightarrow{u}, \overrightarrow{v})$.

Como a função RHH produz um único bit, para criar valores de *hash* mais longos é necessário produzir vários vetores $\overrightarrow{r}$ e concatenar os resultados, compondo uma sequência binária. Para cada posição na sequência binária, vetores similares têm alta probabilidade de produzirem um resultado idêntico. Dessa forma, a distância de Hamming entre os valores de *hash* de vetores similares será pequena.

4.3 Sumário

O volume de dados disponíveis atualmente na WWW cresce a cada momento. Cada vez mais são necessárias formas de se lidar com esse volume heterogêneo de dados, facilitando a busca e melhorando o relacionamento entre aqueles que são considerados similares. Este capítulo apresentou o conceito de funções LSH, funções *hash* que geram valores que mantêm a similaridade dos elementos utilizados na função, essa similaridade definida por uma função de similaridade. Foram apresentados dois exemplos de função LSH: *Min-wise* e o RHH. Uma característica em comum aos trabalhos encontrados na literatura, citados no Capítulo 1, é o fato de todos considerarem somente o conteúdo dos dados para medir a similaridade entre eles. No próximo capítulo será apresentada a principal contribuição deste trabalho que permite relacionar conceitualmente dados de diferentes tipos, inovando a abordagem tradicional na qual a similaridade é restrita ao conteúdo dos dados, como comentado anteriormente.

Capítulo 5

Utilização das funções LSH a partir da classificação conceitual dos dados

Como apresentado nos capítulos anteriores, devido ao crescimento da Web Semântica e também por consequência do volume de dados semanticamente anotados disponíveis na Web, a busca conceitual se torna cada vez mais importante, pois pode reduzir o espaço da busca e comparar dados de diferentes tipos. É por meio desse tipo de consulta que a manipulação e uso desse grande volume de dados se torna factível.

Como já dito nesta tese, diferentes tipos de dados podem ser relacionados entre si a partir do ponto de vista de um domínio de conhecimento que os classifica. Essa classificação pode ser usada no momento da indexação desses dados, mantendo a relação que esses possuem na sua própria indexação, facilitando a busca conceitual.

Neste capítulo será apresentado como foi utilizada a classificação conceitual dos dados (Capítulo 2) em funções LSH (Capítulo 4). Foram testados diferentes métodos de medida de similaridade entre conceitos de uma mesma ontologia (Capítulo 3).

5.1 Geração de *hash* LSH a partir de ontologias *lightweight*

Uma maneira de se representar a relação entre os conceitos de uma estrutura hierárquica, tal qual uma ontologia simples, é por meio de vetores. Dessa forma, cada conceito pode ser representado por um vetor que reflita a sua similaridade com os outros conceitos pertencentes à mesma ontologia.

Dentre as funções de *hash* LSH apresentadas no Capítulo 4, destaca-se a *Random Hyperplane Hash* (RHH), concebida para receber dados representados por vetores e manter a similaridade definida pelo cosseno do ângulo entre esses vetores. A função RHH, portanto, torna-se a escolha natural para este trabalho.

Na metologia proposta, parte-se de uma ontologia de domínio e, utilizando algum método para medir a similaridade entre conceitos de uma ontologia, como os apresentados no Capítulo 3, cria-se vetores conceituais para cada conceito dessa ontologia. Esses vetores são utilizados na função RHH, gerando assim valores de *hash* que mantenham a similaridade entre os conceitos, tal similaridade definida por um método. A Figura 5.1 ilustra o procedimento.

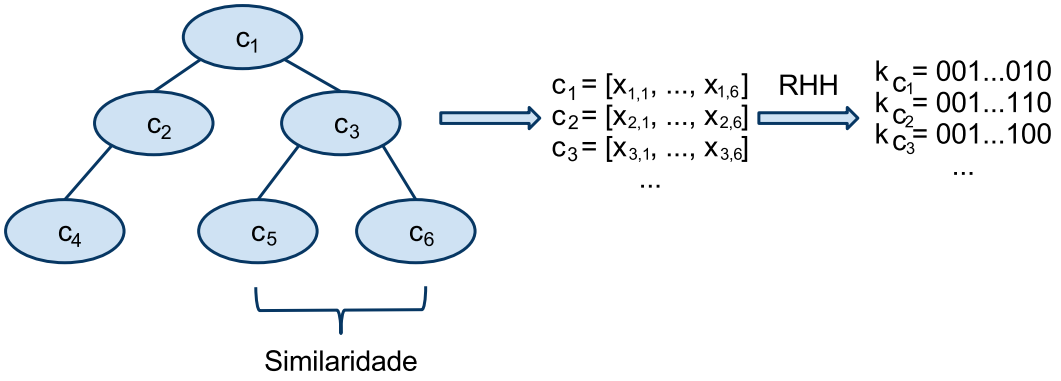


Fig. 5.1: Criação dos identificadores conceituais utilizando uma função LSH.

Neste ponto, portanto, é necessária a criação de vetores de conceito para cada conceito da ontologia, seguindo um determinado método de similaridade. A próxima subseção apresenta esse procedimento.

5.1.1 Geração de *hash* utilizando vários métodos de similaridade entre conceitos de uma ontologia

Entre os métodos apresentados no Capítulo 3, aquele baseado no algoritmo utilizado pelo eTVSM [1] constrói um vetor de conceito para cada conceito de uma ontologia, sendo o cosseno do ângulo entre esses vetores a medida de similaridade. Para os outros métodos, os quais já determinam o valor da similaridade entre dois conceitos, faz-se necessária a construção do vetor de conceito. Nesse caso, definiu-se que, para cada conceito c_i, fosse criado um vetor de conceito da seguinte forma:

$$\overrightarrow{v_{c_i}} = (sim(c_i, c_0), sim(c_i, c_1), ..., sim(c_i, c_j))$$

no qual $\overrightarrow{v_{c_i}}$ representa o vetor de conceito do conceito c_i e $sim(c_i, c_j)$ é a similaridade entre o conceito c_i e c_j definida por um dos métodos de similaridade entre conceitos de uma ontologia.

No caso do método *Bouquet*, esse vetor é composto pelo tamanho do menor caminho entre os conceitos. Para criar um vetor contendo valores dentro do intervalo [0, 1], os vetores com os valores dos menores caminhos são normalizados pelo valor do maior caminho entre todos os menores caminhos. Portanto, para cada elemento $Bouquet(c_i, c_j)$, que representa o menor caminho entre c_i e c_j, foi computado um valor $sim_{Bouquet}(c_i, c_j)$ definido por

$$sim_{Bouquet}(c_i, c_j) = 1 - \frac{Bouquet(c_i,c_j)}{max\{Bouquet(c_1,c_2),...,Bouquet(c_n,c_m)\}}$$

para n e m variando até o número máximo de conceitos da ontologia. Essa transformação é necessária devido ao fato de que, por esse método, conforme a distância diminui, a similaridade entre os conceitos aumenta.

Portanto, para uma ontologia *lightweight*, a função de *hash* LSH para os conceitos pode ser resumida nos seguintes passos, apresentados também na Figura 5.1:

- são gerados vetores de conceito para cada conceito de uma ontologia: se a criação dos vetores seguir o algoritmo do eTVSM [1], como mostrado na Seção 3.2, esses próprios vetores podem ser utilizados para o cálculo do *hash*. Se for utilizado algum outro método de similaridade, como os apresentados no Capítulo 3, os vetores são formados como mostrado nesta seção;

- os vetores de conceito são utilizados na função RHH: como mostrado na Seção 4.2.2, são criados *n* vetores $\overrightarrow{r}$, cada coordenada desses sorteada de uma distribuição Normal (0, 1), e é calculado o produto escalar entre esses vetores $\overrightarrow{r}$ e os vetores de conceito. Os resultados dos *n* produtos escalares são concatenados gerando uma cadeia de *n* bits. Essa função mantém a similaridade do cosseno entre os vetores de conceito.

De posse dos vetores de conceito dos métodos, devido ao fato da função utilizada (RHH) manter a similaridade do cosseno do ângulo entre esses vetores, é preciso calcular a correlação[1] entre o valor do cosseno e os valores retornados pelos métodos para cada par de conceitos. Em outras palavras, é necessário checar se há correlação entre $sim(c_i, c_j)$ e $\cos\theta(\overrightarrow{v_{c_i}}, \overrightarrow{v_{c_j}})$ para cada um dos métodos. Não é preciso checar a correlação dos vetores gerados pelo eTVSM pois, por definição do método, é exatamente o valor do cosseno do ângulo entre os vetores que define a similaridade entre os conceitos. Para calcular a correlação no caso dos outros métodos criou-se, para cada conceito, um vetor $\overrightarrow{v_{\cos(c_i)}}$ formado por:

$$\overrightarrow{v_{\cos(c_i)}} = (\cos\theta(\overrightarrow{v_{c_i}}, \overrightarrow{v_{c_0}}), \cos\theta(\overrightarrow{v_{c_i}}, \overrightarrow{v_{c_1}}), ..., \cos\theta(\overrightarrow{v_{c_i}}, \overrightarrow{v_{c_j}}))$$

e foi calculada a correlação entre $\overrightarrow{v_{\cos(c_i)}}$ e $\overrightarrow{v_{c_i}}$. Foram utilizadas duas ontologias: ontologia 1 com 7 conceitos como mostrada na Figura 3.3; e ontologia 2, em forma de árvore binária com 127 conceitos. O intuito de se utilizar duas ontologias diferentes com dois tamanhos diferentes é determinar se o tamanho da ontologia influencia na correlação.

A Tabela 5.1.1 apresenta a média da correlação entre o cosseno do ângulo entre os vetores e o valor da similaridade definida pelo método, para cada par de conceitos das ontologias.

[1]Em Probabilidade e Estatística, correlação, também conhecida como coeficiente de correlação, indica a força e a direção do relacionamento linear entre duas variáveis aleatórias. Nesta tese foi utilizada a correlação de Pearson que

Tab. 5.1: Média da correlação entre os métodos e o cosseno do ângulo entre os vetores.

Método	Ontologia 1	Ontologia 2
Bouquet	0.93	0.91
Leacock	0.94	0.89
Wu	0.96	0.91
Li	0.94	0.89

Em todos os métodos, a correlação entre os valores de similaridade e o cosseno do ângulo entre os vetores é alta. Esse resultado indica que a similaridade dos métodos correlaciona-se positivamente com o cosseno do ângulo entre os vetores, valor que será mantido pelo RHH.

A próxima subseção apresenta a correlação entre os métodos a os identificadores binários gerados (chaves de indexação).

5.1.2 Correlação entre métodos e chaves geradas

Esta seção mostra a correlação entre os valores binários gerados pela função RHH e o valor da similaridade definida pelos métodos. É esperado que as chaves criadas reflitam a similaridade definida pelos métodos na sua distância de Hamming. Conceitos mais similares devem possuir chaves com baixa distância de Hamming.

Para cada método, foram criados vetores de conceitos e esses foram utilizados no RHH, seguindo o mesmo procedimento detalhado no Capítulo 4, criando chaves de 128 bits, uma para cada conceito.

Após criadas as chaves para cada conceito de cada uma das ontologias, foi calculada a correlação entre a similaridade dada pelos métodos e a similaridade de Hamming entre as chaves dos conceitos. Para isso foi criado, para cada conceito c_i, um vetor $\overrightarrow{h_{c_i}}$, definido por:

$$\overrightarrow{h_{c_i}} = (sim_{ham}(k_{c_i}, k_{c_0}), sim_{ham}(k_{c_i}, k_{c_1}), ..., sim_{ham}(k_{c_i}, k_{c_j}))$$

no qual k_{c_i} é o resultado do RHH para o vetor do conceito c_i e $sim_{ham}(k_{c_i}, k_{c_j})$ representa a similaridade de Hamming entre as chaves k_{c_i} e k_{c_j} calculada da seguinte forma:

$$sim_{ham}(k_{c_i}, k_{c_j}) = \frac{\textit{número de bits coincidentes}}{\textit{tamanho das chaves}}.$$

Ou seja, para duas sequências binárias distintas, de mesmo tamanho, é contado o número de bits em que elas coincidem e divide-se esse valor pelo tamanho das chaves.

Foi então calculada a correlação entre os valores de $\overrightarrow{v_{c_i}}$ e $\overrightarrow{h_{c_i}}$ para todos os conceitos das ontologias. A Tabela 5.1.2 apresenta os resultados.

retorna valores no intervalo [-1, 1], sendo que o valor -1 indica uma correlação negativa perfeita entre as duas variáveis,

Tab. 5.2: Correlação entre os métodos e a similaridade de Hamming para as chaves geradas para cada conceito.

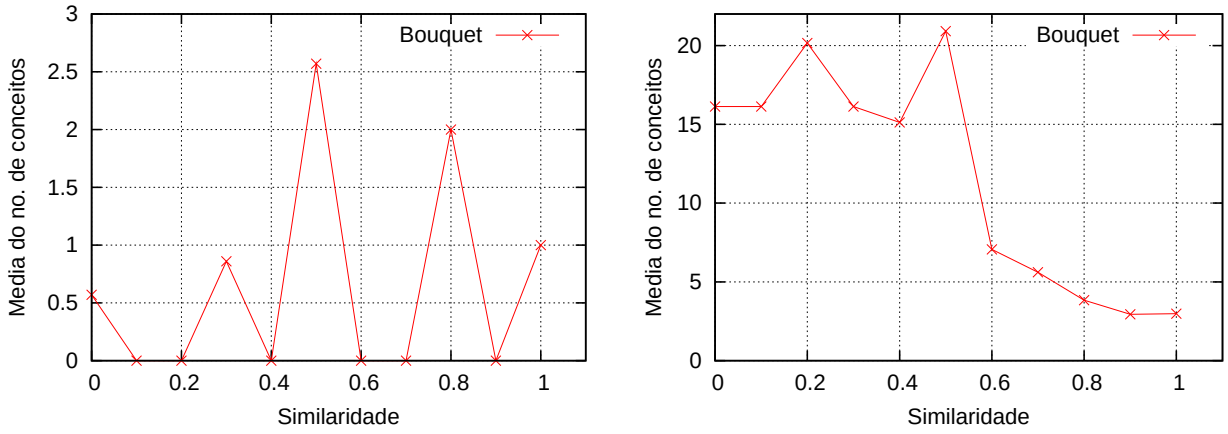
Método	Ontologia 1	Ontologia 2
Bouquet	0.93	0.91
Leacock	0.94	0.89
Wu	0.96	0.91
Li	0.94	0.89
eTVSM	0.96	0.93

Como visto na tabela, a correlação entre os métodos e a similaridade de Hamming das chaves criadas é alta. Dessa forma é possível concluir que a similaridade dos métodos é mantida na geração dos valores retornados pela função RHH. O resultado confirma o que era esperado: como a correlação entre os métodos e o cosseno do ângulo entre os vetores formados pelos valores definidos por esses métodos é forte, apresentada na Tabela 5.1.1, e o RHH mantém a similaridade referente ao cosseno do ângulo entre os vetores, era de se esperar que houvesse uma correlação alta na comparação entre a distância de Hamming das chaves geradas e os valores definidos pelos métodos. Essa característica também é confirmada pelo Lema 3, apresentado na Seção 4.1, que define que funções de similaridade pode ser incorporada em um cubo de Hamming.

As figuras a seguir mostram graficamente, para os diferentes métodos, esse comportamento. Os métodos *Bouquet*, *Leacock*, *Wu*, *Li* e *eTVSM* são apresentados, respectivamente. Para todos os métodos, para as duas ontologias, é mostrada a relação entre a média do número de conceitos e o nível de similaridade definido pelos métodos, ou seja, cada ponto no gráfico representa, em média, quantos conceitos existem naquele nível de similaridade. Exemplo: na Figura 5.2(a), existem, em média, 2 conceitos no intervalo (0,7, 0,8] de similaridade para cada conceito da ontologia. Em seguida, é mostrada a relação entre a média do número de chaves geradas para os conceitos e a similaridade de Hamming entre elas. Exemplo: para o teste mostrado na Figura 5.3(b), em média, há 2 chaves de conceitos com similaridade de Hamming de 0,6 para todas as outras.

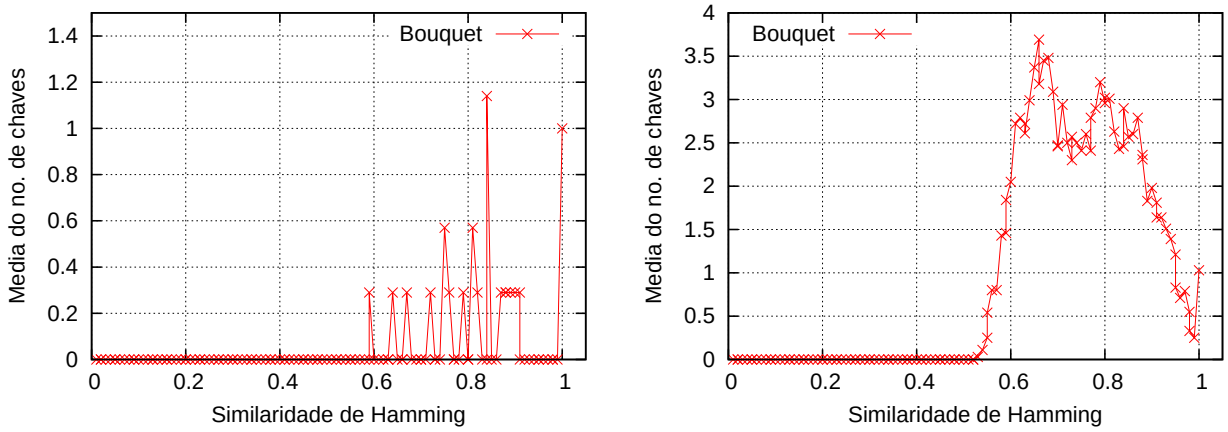
ou seja, uma cresce quando a outra diminui; o valor 1 indica uma correlação positiva perfeita entre as duas variáveis, ou seja, ambas crescem ou diminuem em concordância e o valor 0 indica que não há dependência linear entre as variáveis.

Método *Bouquet*:



(a) Ontologia 1 (b) Ontologia 2

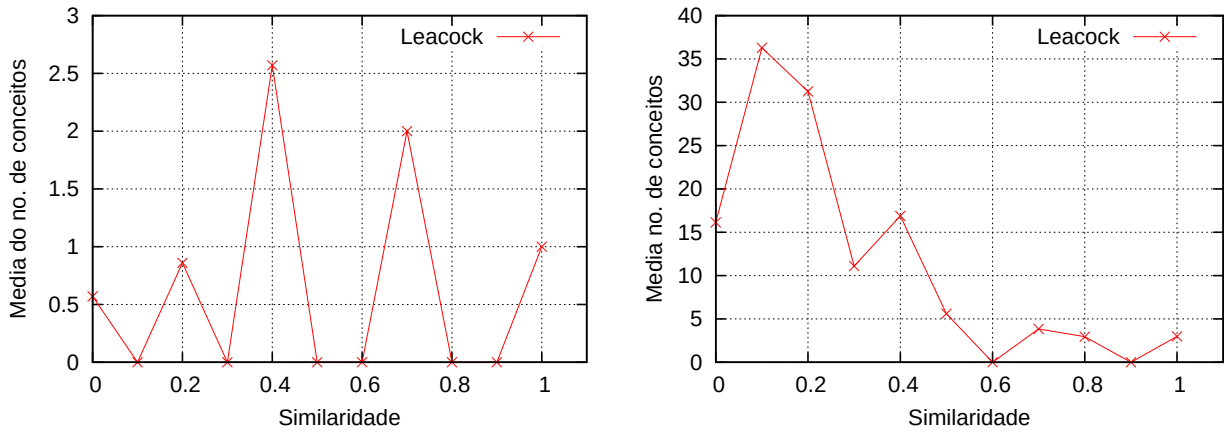
Fig. 5.2: Conceitos por similaridade para o método *Bouquet*.



(a) Ontologia 1 (b) Ontologia 2

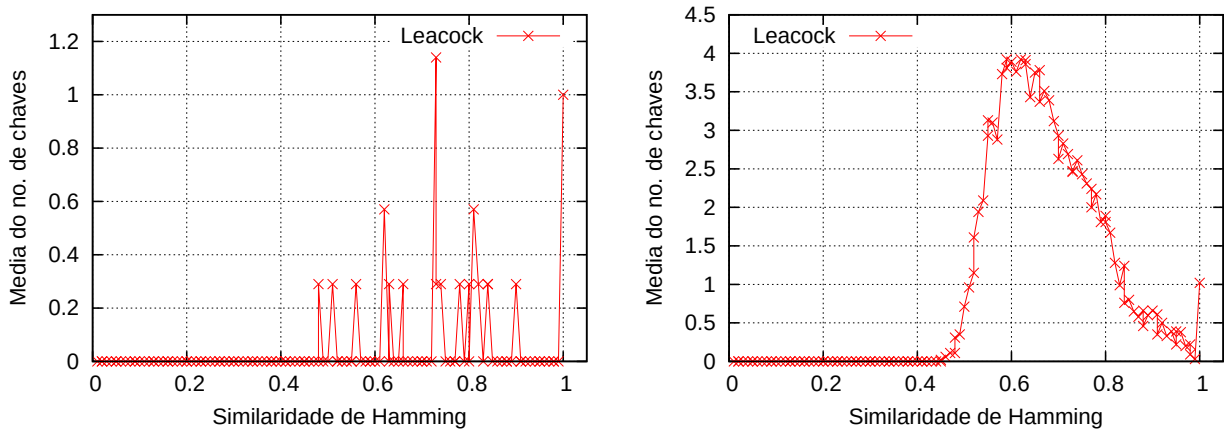
Fig. 5.3: Chaves por similaridade de Hamming para o método *Bouquet*.

Método *Leacock*:



(a) Ontologia 1 (b) Ontologia 2

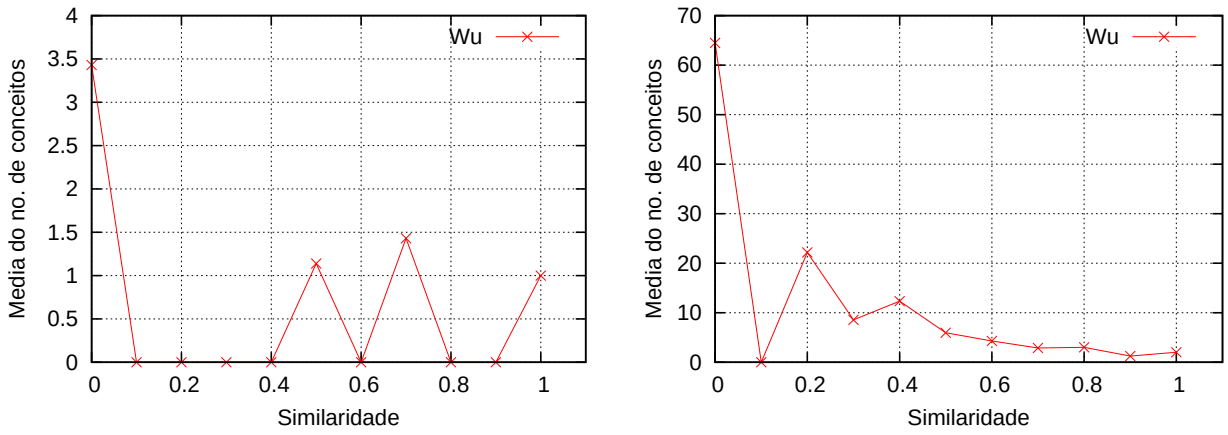
Fig. 5.4: Conceitos por similaridade para o método *Leacock*.



(a) Ontologia 1 (b) Ontologia 2

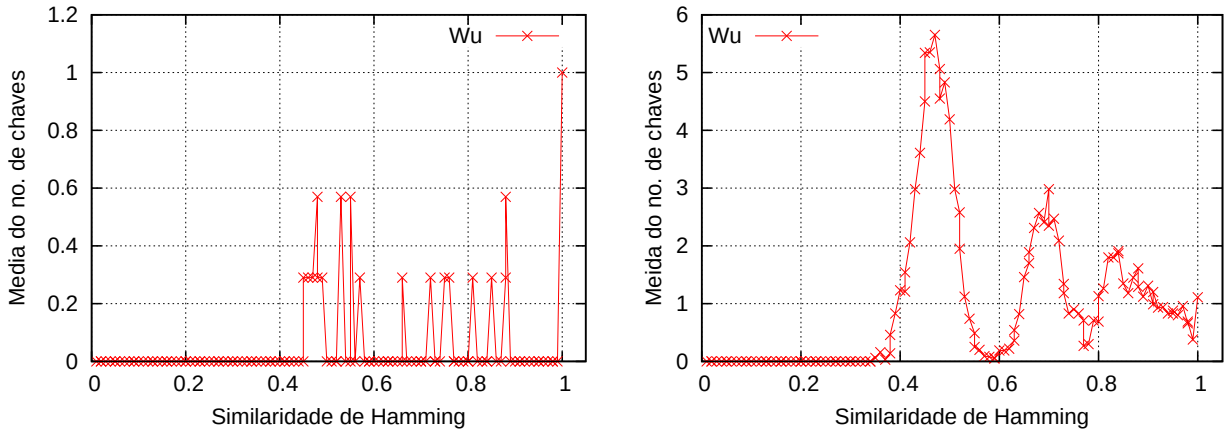
Fig. 5.5: Chaves por similaridade de Hamming para o método *Leacock*.

Método *Wu*:



(a) Ontologia 1 (b) Ontologia 2

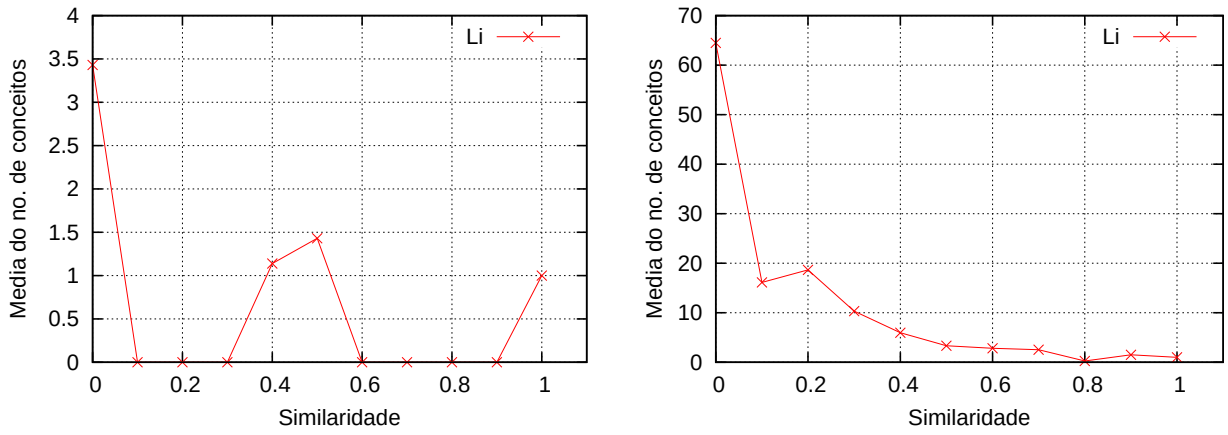
Fig. 5.6: Conceitos por similaridade para o método *Wu*.



(a) Ontologia 1 (b) Ontologia 2

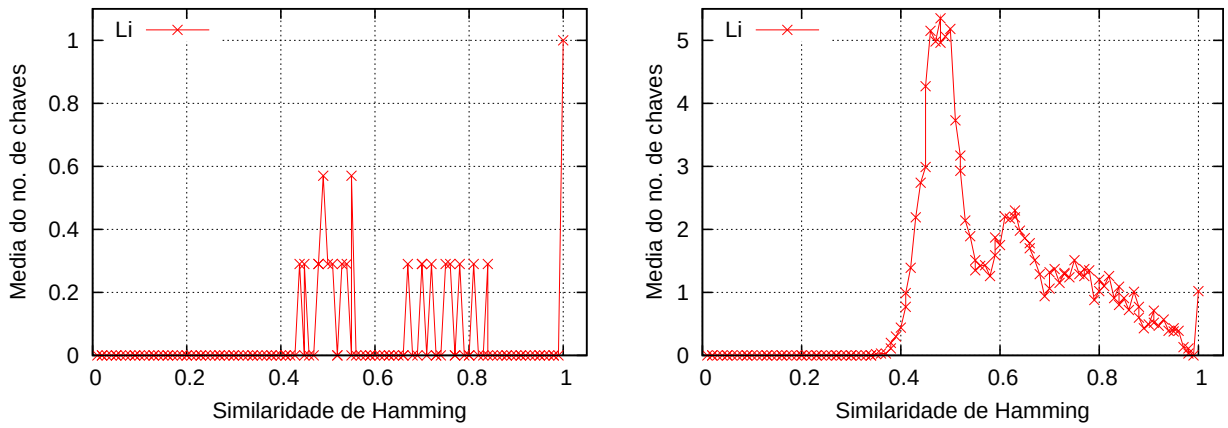
Fig. 5.7: Chaves por similaridade de Hamming para o método *Wu*.

Método *Li*:



(a) Ontologia 1 (b) Ontologia 2

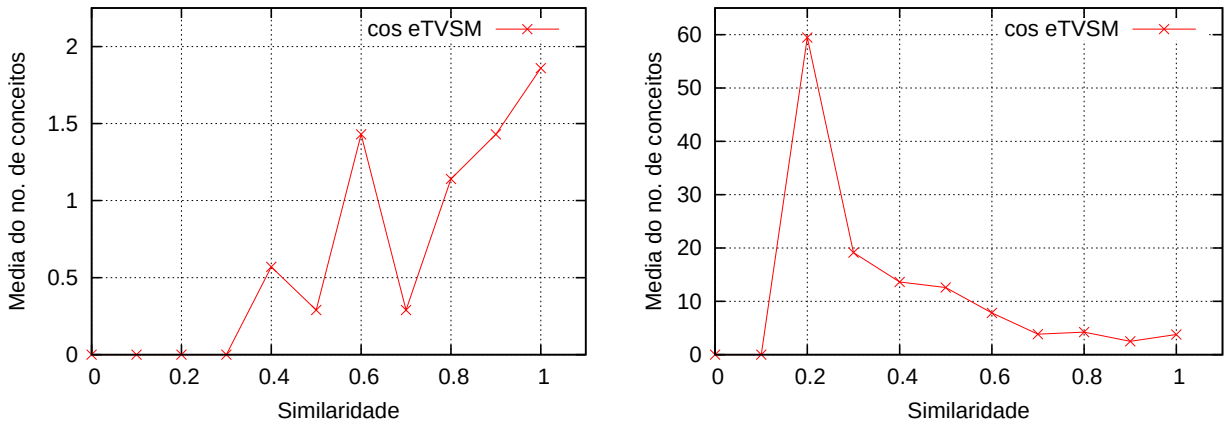
Fig. 5.8: Conceitos por similaridade para o método *Li*.



(a) Ontologia 1 (b) Ontologia 2

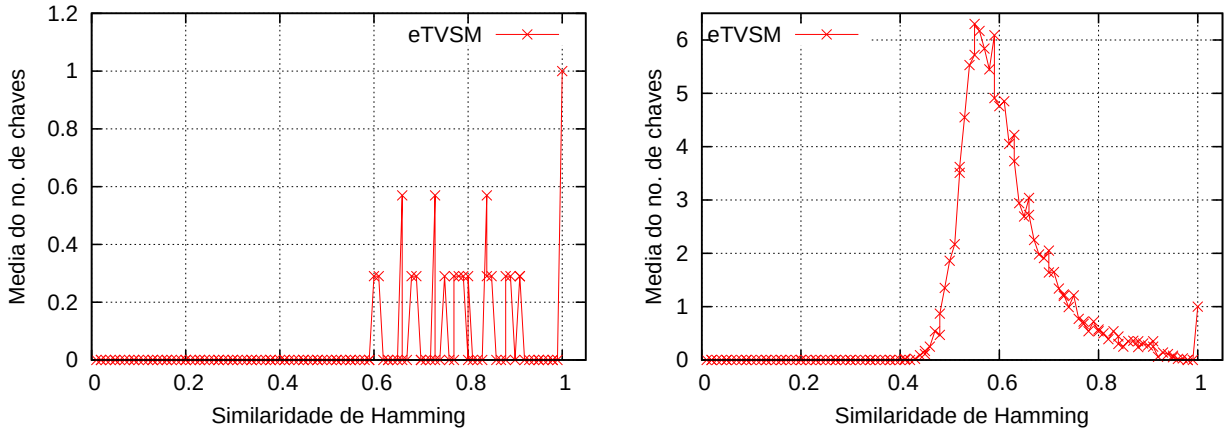
Fig. 5.9: Chaves por similaridade de Hamming para o método *Li*.

Método *eTVSM*:



(a) Ontologia 1 (b) Ontologia 2

Fig. 5.10: Conceitos por similaridade para o cosseno do ângulo entre os vetores do método *eTVSM*.



(a) Ontologia 1 (b) Ontologia 2

Fig. 5.11: Chaves por similaridade de Hamming para o cosseno do ângulo entre os vetores do método *eTVSM*.

As figuras mostram que os gráficos são proporcionais, ou seja, aproximadamente, o número de conceitos para determinados níveis de similaridade é proporcional ao número de chaves para determinados níveis definidos pela similaridade de Hamming.

É possível notar nos gráficos que mostram o número de chaves pela similaridade de Hamming que não há chaves que se encontram entre 0 e aproximadamente 0,5 de similaridade. A explicação para isso é que, mesmo que o método de similaridade utilizado defina que dois desses conceitos possuam 0 de similaridade entre si, ainda assim esses possuem 50% de chances de que, para um determinado bit, possuam o mesmo valor. Em outras palavras, 0 de probabilidade resulta em 50% de chances de

coincidirem no resultado de um bit no *hash*. Conforme a probabilidade aumenta, a similaridade de Hamming também aumenta, a partir de aproximadamente 0,5. Se forem comparadas, por exemplo, *n* chaves geradas de forma totalmente aleatória (por exemplo, utilizando um *hash* tradicional, como o MD5), em média, a similaridade de Hamming entre todas essas chaves será de aproximadamente 0,5, ou seja, 50% dos bits serão coincidentes. Essa característica é confirmada pelo Lema 2 que mostra as probabilidades para funções LSH que mapeiam objetos nos valores $\{0, 1\}$, como mostrado na Seção 4.1.

Isso mostra que é possível, para qualquer um dos métodos, a criação de identificadores conceituais, utilizando funções LSH, que mantenham a similaridade definida para os pares de conceitos.

5.2 Sumário

As funções de *hash* LSH tem como característica a geração de valores que mantêm a similaridade entre os dados, a partir de uma função de similaridade. Vários trabalhos encontrados na literatura utilizam funções LSH para relacionar diferentes dados, do ponto de vista do seu conteúdo. Por outro lado, existem diversos trabalhos que calculam a similaridade entre conceitos de uma mesma ontologia, mas sem utilizá-la em funções LSH. Este capítulo propôs uma maneira de se utilizar funções LSH para relacionar dados do ponto de vista conceitual a partir de conceitos de ontologias *lightweight*. Foram testados, com sucesso, alguns tipos de funções de similaridade para conceitos de uma mesma ontologia. O próximo capítulo apresenta os cenários de aplicação, alguns testes e seus resultados.

Capítulo 6

Testes e análise

Como visto no decorrer desta tese, devido ao grande volume de dados atualmente disponível na WWW, novas estratégias são necessárias para se lidar com toda essa informação, dentre elas pode ser citada a utilização de funções LSH para indexar dados similares de forma a facilitar a sua busca. Uma característica dos trabalhos encontrados na literatura é a falta de comparação conceitual, utilizando funções LSH, entre os dados, existindo geralmente só a comparação em relação ao conteúdo. Esta tese apresentou, no capítulo anterior, um método de criação de funções LSH que relacionam dados no nível conceitual. Dessa forma, diferentes tipos de dados, que representem o mesmo conceito, podem ser relacionados entre si, visando a busca conceitual.

Neste capítulo serão apresentados dois cenários de aplicação, nos quais a utilização de chaves conceituais, geradas por funções LSH, a partir de ontologias, facilita a busca conceitual. Na próxima subseção serão definidos os conceitos básicos dos cenários e em seguida será mostrado cada um deles em particular.

6.1 Conceitos básicos

Antes de apresentar uma proposta de organização de dados conceitualmente classificados é preciso definir, para o escopo dos cenários que serão apresentados, alguns conceitos básicos:

- **dado**: corresponde a qualquer estrutura que possa ser armazenada e recuperada em um sistema computacional. Imagens, textos, áudio, páginas *web*, etc, são exemplos de dados;
- **metadado**: anotação atrelada aos dados que definem a sua classificação a partir de uma hierarquia de conceitos, taxonomia, ontologia, etc e, geralmente, são escritos em RDF ou RDF-S. Neste texto, será usado o termo dado tanto para o dado em si, quanto para o seu metadado, salvo quando indicado;

- **repositório de dados**: toda máquina (ou conjunto de máquinas) que contenha dados ou metadados, de diversos tipos, semanticamente anotados segundo uma determinada classificação com a intenção de compartilhá-los. Exemplos possíveis: máquinas pessoais com arquivos de música e vídeo, grupos de servidores de dados de uma universidade com dados multimídia, repositórios de arquivos RDF da Web Semântica, etc.

- **ontologia**: no contexto desta tese, e como visto no Capítulo 2, ontologia consiste em um esquema de dados que define um domínio formado por conceitos e suas relações, possibilitando o raciocínio e a inferência sobre certo termo da ontologia em relação a outros termos [1]. Por meio de uma ontologia é possível a agregação de termos similares ou mesmo a especialização de um termo em outros. Utilizaremos ontologias simples, também conhecidas na literatura por ontologias *ligthweight*, hierarquia de conceitos ISA, dentre outros nomes, na qual as relações entre conceitos são do tipo "é um" ou "parte de". No decorrer do texto, a palavra *ontologia* fará referência a esse tipo de estrutura.

Esses elementos são utilizados nos cenários de aplicação voltados para a busca conceitual. A próxima subseção descreve como é feita a indexação e a busca dos dados.

6.1.1 Indexação e busca

Nos cenários idealizados nesta tese, a indexação distribuída é feita em uma rede *peer-to-peer* (P2P) estruturada, comumente empregada no compartilhamento de grandes volumes de dados. Geralmente, as redes P2P estruturadas são construídas sobre *Distributed Hash Tables* (DHT), as quais podem ser organizadas de diversas formas do ponto de vista topológico. Nesta tese foram consideradas DHTs baseadas no *Chord* [17], que organizam os nós em um anel virtual, topologia comum entre as redes P2P estruturadas disponíveis na literatura. No anel, cada nó é responsável pela indexação das chaves que pertençam ao espaço delimitado entre o seu próprio identificador e o identificador de seu antecessor. Entre os nós que fazem parte da DHT são criadas ligações (*fingers*) e essas ligações são utilizadas no roteamento de mensagens dentro do anel. Uma das formas mais comuns de indexação em DHTs é através da criação de chaves para identificação de dados, utilizando uma função de *hash* tradicional como, por exemplo, as funções MD5 e SHA-1. O valor obtido pelo *hash* passa a ser o identificador daquele conteúdo na rede P2P, por meio do qual o dado é inserido e recuperado, em um esquema de *put* e *get*. Os dados são inseridos pela primitiva *put(k, v)* e obtidos pela primitiva *get(k)*, nas quais *k* é o identificador do dado e *v* é um valor atrelado ao identificador, podendo ser o dado propriamente dito, um metadado, um localizador de onde o dado pode ser obtido, etc. A Figura 6.1 ilustra esse roteamento.

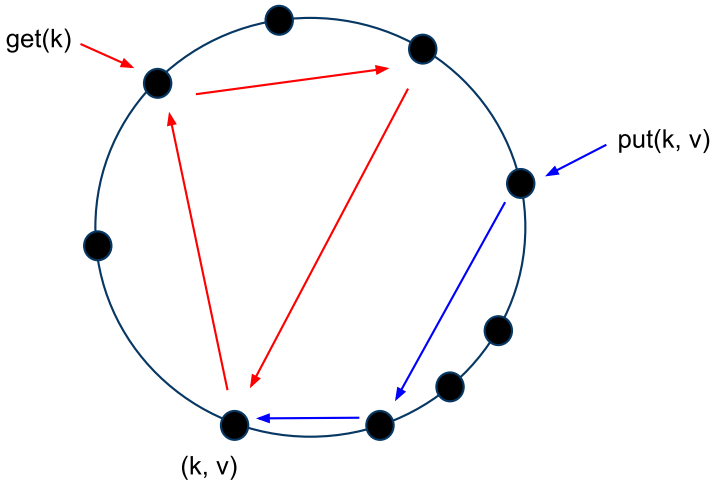


Fig. 6.1: Esquema geral de uma DHT em forma de anel.

Uma característica decorrente da utilização de funções comuns de *hash* na criação dos identificadores dos dados é a total falta de relação semântica entre o valor obtido pelo *hash* e o dado que o originou devido à natureza aleatória de tais funções. Uma alternativa é a utilização de funções *hash* sensíveis à localidade (*Locality Sensitive Hash* - LSH) [81][21]. Como apresentado no Capítulo 4, essas funções, sempre atreladas a uma função de similaridade, geram valores que carregam as características dos dados que os originaram. Dessa forma, dados classificados de forma similar possuem grande probabilidade de gerarem valores de *hash* próximos em uma determinada métrica ou, até mesmo, idênticos.

A indexação, portanto, é feita utilizando funções LSH, de maneira que a similaridade entre os conceitos das ontologias seja levada em consideração. Conceitos similares são indexados próximos uns aos outros no espaço de endereçamento da DHT, por meio de chaves de conceito, facilitando a obtenção de dados relacionados pela redução do número de saltos necessários para obtê-los na DHT. O número de saltos expressa a quantidade de nós que precisam ser visitados para que a mensagem chegue até o seu destino.

Um cenário para uso da busca conceitual, como proposto nesta tese, pode ser ilustrado por um usuário que, ao fazer uma consulta no sistema por meio de um portal de busca, indica o conceito sobre o qual ele queira pesquisar. A determinação desse conceito pode ser feita: (i) de forma explícita, na qual o usuário indica, a partir de uma ontologia obtida de um repositório de ontologias, qual o conceito de interesse para a busca; ou (ii) de forma implícita, na qual o conceito buscado e a ontologia associada são determinados por meio de analisadores de linguagem natural e raciocinadores. As técnicas necessárias para a realização da forma implícita estão além do escopo desta tese. Ainda no momento da consulta, o usuário pode também escolher um nível de similaridade no intervalo [0, 1], o qual ele admite que seja usado na sua busca. Dados classificados em conceitos que possuam

até esse índice de similaridade com o conceito buscado, tal similaridade definida por um método de similaridade entre conceitos de uma ontologia, como vistos no Capítulo 3, podem fazer parte da resposta. Esse valor pode, inclusive, ser intrínseco ao sistema ou até mesmo usado na ordenação dos resultados apresentados (*ranking*). É importante citar que, dependendo de como são estruturadas as ontologias presentes no sistema, alguns dos métodos utilizados para medir a similaridade entre conceitos podem ser mais indicados que outros, como discutido na Seção 3.2.6.

Para a busca na rede P2P, o conceito buscado é traduzido em uma chave de conceito indexada, e são retornadas informações armazenadas com aquela chave, como o localizador dos repositórios que possuem dados classificados naquele conceito ou o próprio metadado indicando a URI de onde obtê-lo. Dados inseridos com chaves de conceitos similares também são retornados, recuperados por uma busca iniciada a partir do nó que armazena a chave do primeiro conceito buscado, respeitando-se o índice de similaridade pedido pelo usuário ou intrínseco ao sistema. É importante citar que nem todo dado relacionado aos conceitos buscados deve necessariamente ser retornado ao usuário. O resultado da busca pode ser utilizado por um outro serviço, responsável pelo tratamento dos dados obtidos, para melhor atender a requisição do usuário. Esse serviço pode, inclusive, encontrar dados semelhantes do ponto de vista de conteúdo entre aqueles classificados em um mesmo conceito.

Em diversos trabalhos encontrados na literatura, como [20] e [24], há a preocupação da manutenção da relação entre dados similares, do ponto de vista de conteúdo, quando armazenados em alguma estrutura de indexação, seja centralizada (bases de dados) ou distribuída (redes P2P baseadas em DHTs, etc). Uma característica da abordagem desses trabalhos é que, como as funções LSH são atreladas à similaridade do conteúdo dos dados para gerar os identificadores para cada dado, e não para um grupo, a busca conceitual é dificultada.

Neste trabalho, visando exatamente a busca conceitual, como apresentada no Capítulo 2, é utilizada a classificação conceitual para gerar uma função LSH atrelada à similaridade entre os conceitos. Dessa forma, conceitos similares são armazenados próximos no espaço virtual de uma rede P2P, facilitando a busca por dados relacionados, como visto na Figura 6.2.

A próxima subseção descreve os cenários de aplicação mostrando como esses podem ser implementados para a distribuição de dados mantendo sua similaridade.

6.1.2 Descrição dos cenários

Nos cenários aqui abordados, diversos repositórios disponibilizam dados multimídia (imagens, textos, áudio, etc) sobre assuntos variados. Isso pode ser exemplificado por um ambiente organizacional de uma empresa, com diversas filiais distribuídas ou um ambiente de uma universidade com diversas faculdades espalhadas por vários campi.

Dentro dessas organizações podem existir vários departamentos, cada qual responsável pela ge-

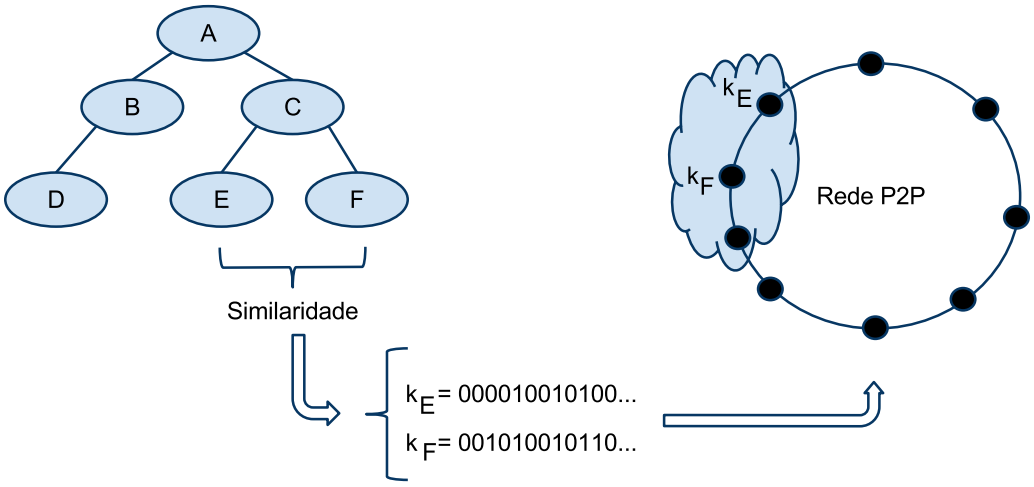


Fig. 6.2: Indexação de conceitos similares em uma rede P2P.

rência de dados classificados em um ou mais domínios de conhecimento. A definição desses domínios de conhecimento é de responsabilidade de cada departamento, sendo feita por especialistas da área. Esses departamentos possuem repositórios que disponibilizam dados de vários tipos, classificados no domínio de conhecimento sob sua responsabilidade. A definição do domínio de conhecimento é feita utilizando ontologias simples com dúzias ou centenas de conceitos. Uma forma comumente utilizada para a formalização de uma ontologia é feita utilizando OWL, uma linguagem, baseada em RDF, específica para a definição e instanciação de uma ontologia. Em OWL é possível descrever conceitos e subconceitos de uma ontologia, assim como a classificação de conteúdos em um desses conceitos. A Figura 6.3 mostra um exemplo de definição de conceitos e subconceitos da ontologia da Figura 3.3 em um arquivo OWL, utilizando as *tags owl:Class* e *owl:subClassOf*. Na figura são definidos os conceitos c_1, c_2 e c_4, indicando que c_2 é subconceito de c_1 e c_4 é subconceito de c_2. É recomendado pelos padrões da Web Semântica que cada conceito descrito remeta a uma URI, possibilitando a obtenção de mais informações a seu respeito. Por exemplo, na Figura 6.3 é indicado que mais informações sobre o conceito c_1 podem ser obtidas em *http://www.exemplo.org/AlgumDominio/C1*.

A classificação de um dado em um conceito de uma ontologia pode ser expressada no seu metadado, comumente escrito em RDF, bastando que esse possua uma tripla indicando essa classificação através da propriedade *rdf:type*[1]. A Figura 6.4 mostra um exemplo de classificação, no qual é indicado que o dado "algumDado", descrito na URI *http://www.exemplo.org/algumDado*, é classificado no conceito c_2 da ontologia.

Seguindo o mesmo princípio, outro cenário possível seria a indexação de dados provenientes de comunidades virtuais ou redes sociais. Existem na literatura algumas iniciativas como o SIOC[2]

[1]Um arquivo RDF pode ser interpretado como uma coleção de triplas do tipo "sujeito, predicado, objeto".
[2]Lê-se como a palavra *shock*. Corresponde também à palavra irlandesa para geada.

```
<owl:Class rdf:about="http://www.exemplo.org/AlgumDominio/C1">
  <rdfs:label>C1</rdfs:label>
  <rdfs:comment>Conceito raiz.</rdfs:comment>
</owl:Class>
...
<owl:Class rdf:about="http://www.exemplo.org/AlgumDominio/C2">
  <rdfs:subClassOf rdf:resource="http://www.exemplo.org/AlgumDominio/C1"/>
  <rdfs:label>C2</rdfs:label>
  <rdfs:comment>Conceito c2.</rdfs:comment>
</owl:Class>
...
<owl:Class rdf:about="http://www.exemplo.org/AlgumDominio/C4">
  <rdfs:subClassOf rdf:resource="http://www.exemplo.org/AlgumDominio/C2"/>
  <rdfs:label>C4</rdfs:label>
  <rdfs:comment>Conceito c4.</rdfs:comment>
</owl:Class>
```

Fig. 6.3: Exemplo trechos de um arquivo OWL.

```
<rdf:Description rdf:about="http://www.exemplo.org/algumDado">
    <rdf:type rdf:resource="http://www.exemplo.org/AlgumDominio/C2"/>
</rdf:Description>
```

Fig. 6.4: Exemplo de um trecho do metadado em RDF de um dado classificado no conceito c_2.

(*Semantically-Interlinked Online Communities*) [86] que propõem padrões e formas de se disponibilizar esse tipo de dado de forma a interligar diferentes comunidades. O SIOC, especificamente, define como disponibilizar, de forma padronizada, dados originados em fóruns, *blogs*, listas de discussão, etc. O SIOC define uma ontologia que classifica esse tipo de dado em conceitos como *Site*, *Forum*, *User Account*, *Post*, entre outros. Uma propriedade definida pelo SIOC que pode ser atribuída a quase todos esses conceitos é chamada *Topic*, sendo que essa define o assunto ou tópico daquele conteúdo e é muito utilizada para comparar dados entre si.

Uma maneira de utilizar o padrão definido pelo SIOC em conjunto com as propostas aqui apresentadas seria estabelecer uma relação entre a propriedade *Topic*, definido pelo SIOC, e um conceito de uma ontologia que define um domínio de conhecimento. Dessa forma, dados de comunidades virtuais, que já possuem uma estruturação e classificação feita pela ontologia do SIOC, terão um outro nível de classificação, dessa vez conceitual. A Figura 6.5 mostra um exemplo de trecho de um arquivo que utiliza o vocabulário definido pelo SIOC e vincula um *Post* (*tag sioc:Post*), que possui um criador definido pelo vocábulo *UserAccount* (*tag sioc:UserAccount*), ao conceito c_2 definido no exemplo da Figura 6.3 (*tag sioc:Topic*).

Essa classificação é vantajosa para a indexação desses dados, pois facilita a busca por conteúdos de comunidades virtuais que são classificados em conceitos similares em um domínio de conhecimento

```
<sioc:Post rdf:about="http://www.exemplo.org/algumPost">
...
    <sioc:has_creator>
        <sioc:UserAccount rdf:about="http://www.exemplo.org/algumUsuario"
              rdfs:label="UsuárioX">
        </sioc:UserAccount>
    </sioc:has_creator>
    <sioc:content>Informação sobre algum conteúdo classificado no
          conceito C2...</sioc:content>
    <sioc:Topic rdf:resource="http://www.exemplo.org/AlgumDominio/C2"/>
...
</sioc:Post>
```

Fig. 6.5: Exemplo de um trecho de um arquivo SIOC.

definido por uma ontologia. Dessa forma, buscas conceituais podem ser feitas procurando instâncias de classes definidas pelo SIOC (*Post*, *Site*, etc) que são relacionadas com o mesmo conceito em um domínio de conhecimento. Essa busca pode ser facilitada caso os dados relacionados estejam indexados próximos na rede P2P.

Por exemplo, seguindo a ontologia do domínio de *Esportes* da Figura 2.3, os dados de comunidades virtuais relacionados com os conceitos desse domínio podem ser indexados em uma mesma região do espaço de indexação facilitando a busca por dados classificados em conceitos similares, a partir de um método de similaridade entre conceitos. Dessa forma, a busca por dados relacionados, inclusive entre comunidades virtuais diferentes é facilitada. Buscas do tipo "obtenha os dados classificados como fóruns das comunidades existentes no sistema que possuem dados classificados no conceito *Futebol* e seus similares" passam a ser viáveis mesmo com os dados indexados de forma distribuída. Em um primeiro momento é identificada a região que indexa aquele domínio de conhecimento (identificado pela ontologia definida em um arquivo OWL) e, dentro da região, é possível executar uma busca por todos aqueles dados que são indicados como sendo do tipo *sioc:Forum* e que possuem como *sioc:Topic* o conceito *Futebol*. Um nível de similaridade pode ser usado para recuperar também aqueles fóruns que possuem dados do tipo *Coletivo* ou mesmo outros esportes. A Figura 6.6 ilustra esse cenário. A indexação dentro de uma região será discutida na Seção 6.6.

A partir da descrição formal das ontologias[3] utilizadas, por exemplo em um arquivo OWL, é possível a utilização de *parsers* para obter a topologia da ontologia e utilizá-la na criação dos valores de *hash* para cada conceito. A classificação do dado na ontologia vincula-o a um conceito, podendo ser utilizada por um repositório no momento da definição conceitual dos dados que possui.

Duas variações de cenário de aplicação serão apresentadas: a primeira indexa as chaves em um

[3]O processo e as técnicas utilizadas na criação de ontologias, assim como na classificação de dados em conceitos de ontologias, amplamente encontradas na literatura, estão fora do escopo desta tese.

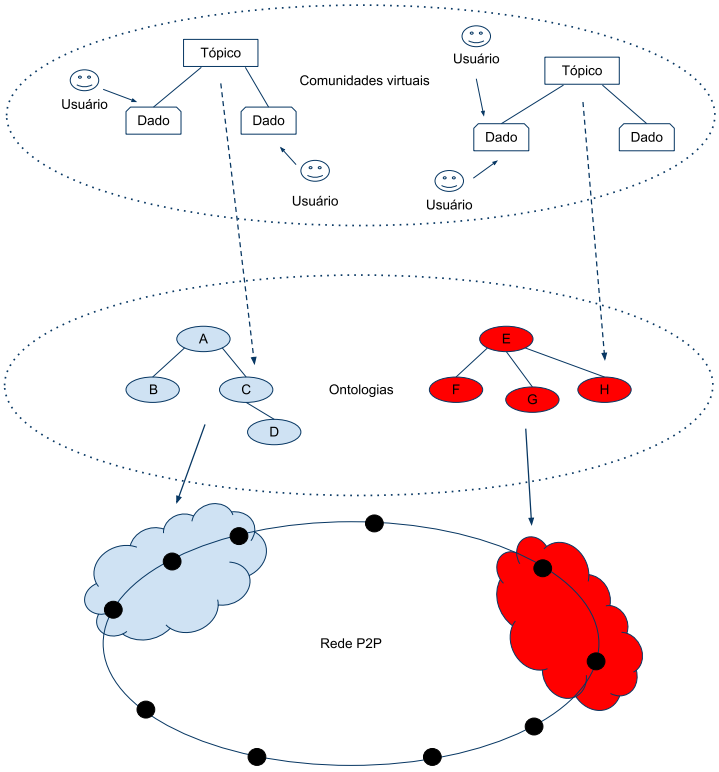


Fig. 6.6: Exemplo de indexação de dados de comunidades virtuais levando em consideração a similaridade conceitual.

cenário com uma DHT convencional, no qual é necessária a agregação das chaves no espaço Euclidiano e a segunda mostra a indexação das chaves em uma DHT que utiliza a distância de Hamming como métrica. Será mostrado que, para o primeiro cenário, é preciso uma alteração na função de *hash* LSH apresentada no Capítulo 5. Devido à falta de trabalhos na literatura que abordam a indexação de dados de forma conceitual, utilizando funções LSH, esta tese não contém testes comparativos com outros trabalhos da literatura.

6.2 Cenário que utiliza a distância Euclidiana

A Figura 6.7 ilustra a organização dos dados em uma DHT Chord visando um esquema de busca conceitual.

A DHT ilustrada na Figura 6.7 indexa os dados em um espaço Euclidiano, ou seja, a distância entre os nós é medida nessa métrica. Dessa forma, para que a busca conceitual seja facilitada, é preciso agregar as chaves nesse espaço.

Como visto no Capítulo 5, a função de *hash* lá apresentada mantém a similaridade das chaves na distância de Hamming. Uma característica em relação à distância de Hamming é que duas chaves

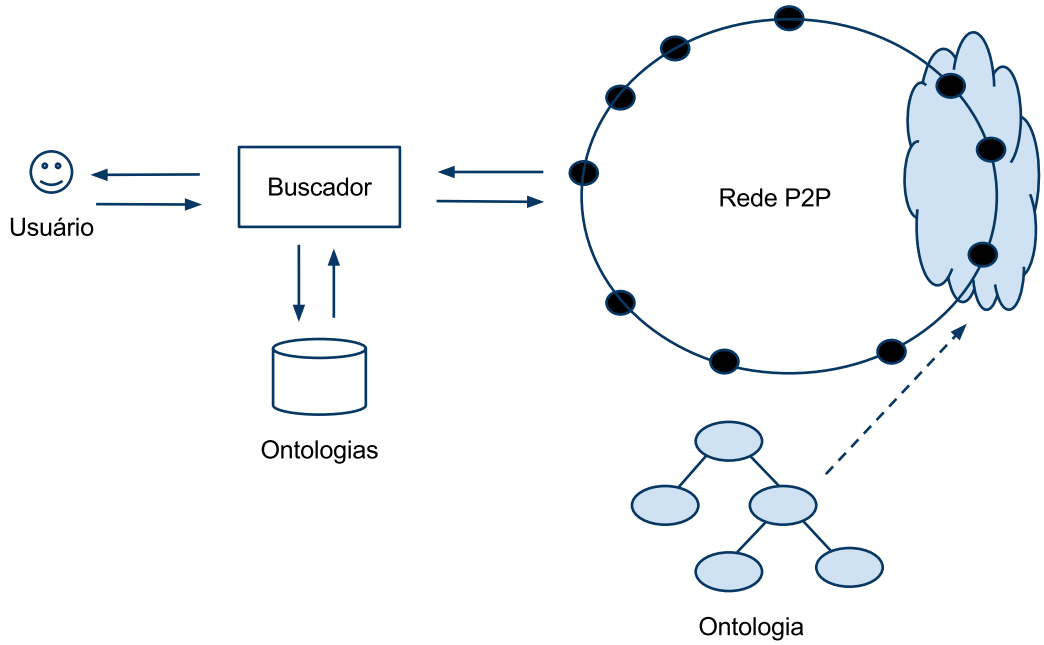


Fig. 6.7: Organização do sistema

podem ser similares nessa distância, mas a indexação dessas chaves em um espaço circular Euclidiano pode resultar em uma grande distância entre elas caso os bits que não coincidem sejam da parte mais significativa do identificador. Por exemplo, os identificadores 00000001 e 10000000 são muito similares do ponto de vista da distância de Hamming, mas em um espaço binário circular seriam indexados em posições totalmente opostas. Isso ocorre devido ao fato do primeiro bit mais significativo dividir o espaço de indexação em duas partes (a primeira metade para identificadores com o primeiro bit igual a “0” e a segunda para aqueles com o primeiro bit igual a “1”); os dois primeiros bits dividem o espaço em 4 partes (dois primeiros bits iguais a “00”, “01”, “10” e “11”), e assim por diante. A Figura 6.8 representa essa situação. É preciso, então, criar identificadores que estejam agregados nessa métrica, como apresentado a seguir.

6.2.1 Criação dos identificadores

A geração de chaves de conceito é feita extraindo-se vetores de conceito para cada conceito da ontologia e aplicando a função RHH (Subseção 4.2.2) nesses vetores. Para o RHH, são gerados n vetores $\overrightarrow{r}$, sendo cada coordenada destes vetores sorteada de uma distribuição Normal com média 0 e variância 1. Uma chave de n bits é obtida para cada conceito pelo produto escalar dos vetores $\overrightarrow{r}$ com os vetores de conceito. Os n resultados são concatenados compondo a chave.

Como apresentado na Subseção 4.2.2, duas chaves de conceito têm probabilidade p de possuírem o mesmo valor para os bits em uma mesma posição, sendo p determinada pela Equação 4.4. Esse

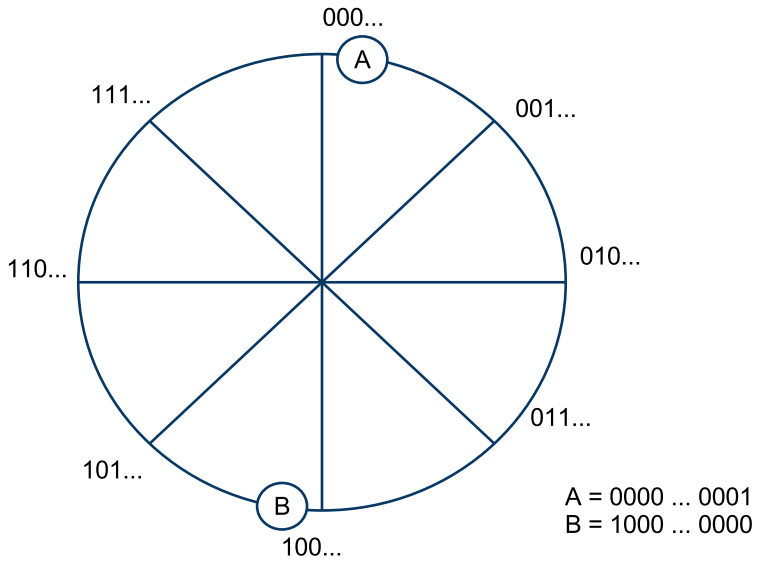


Fig. 6.8: Indexação em um anel.

fato faz com que, mesmo para duas chaves de conceitos muito similares, exista a probabilidade de serem armazenadas distantes uma da outra no espaço virtual de uma DHT. Isso acontecerá se pares de bits, em uma mesma posição, da parte mais significativa dos identificadores não possuírem o mesmo valor. Tentando reduzir esse impacto, a estratégia proposta nesta tese consiste em gerar *m* chaves para cada conceito e usar a que possui o menor valor quando convertido para decimal. Dessa forma, escolhemos a chave que possui o maior prefixo com "0"s como chave do conceito. A probabilidade é alta de que, para conceitos similares, a menor chave criada seja a mesma ou com valores próximos. Isso pode ser explicado pela similaridade de Jaccard usada na função de *hash* LSH *Min-wise* (Capítulo 4): cada grupo de *m* identificadores de conceito pode ser visto como um conjunto de inteiros, fazendo com que a probabilidade de conceitos que possuam o mesmo menor valor seja a mesma explicada na Subseção 4.2.1. Conforme o funcionamento das DHTs tradicionais, nas quais um nó é reponsável por um intervalo de chaves, para que dados similares sejam indexados próximos uns dos outros, somente é preciso que possuam a parte mais significativa do identificador iguais.

Vários trabalhos na literatura, como [20, 19], ao utilizarem funções LSH, indexam cada dado várias vezes na DHT. Isso é feito na tentativa de aumentar a probabilidade de dois identificadores de conceitos similares serem armazenados próximos. Considerando o aumento do volume de dados semanticamente anotados na *Web*, essa abordagem não é a melhor, devido à sobrecarga de identificadores no sistema, mesmo para identificadores de conceitos. A proposta apresentada nesta tese, para este cenário, consiste em criar vários identificadores para cada conceito, mas utilizar somente o menor deles na indexação, não propiciando uma sobrecarga no sistema, à medida que o número de conceitos indexados aumenta. Essa proposta de função de *hash* LSH pode ser definida da seguinte

forma, apresentada na Figura 6.9:

- para cada conceito de uma ontologia *lightweight*, é extraído um vetor de conceito, como apresentado no Capítulo 5, utilizando algum dos métodos de similaridade apresentados no Capítulo 3;
- *m* grupos de *n* vetores $\overrightarrow{r}$ são criados, sendo cada coordenada sorteada de uma distribuição Normal(0, 1);
- para cada vetor de conceito, é aplicada a Função 4.3 (Seção 4.2.2) para cada vetor $\overrightarrow{r}$ de um grupo, gerando uma chave de *n* bits. Esse processo é feito *m* vezes, uma para cada grupo de *m* vetores. Isso equivale a executar a função RHH (Função 4.3 da Seção 4.2.2) *m* vezes;
- a menor chave dentre as *m* geradas é escolhida como chave daquele conceito.

Quanto maior o valor de *m*, mais próximos os conceitos são armazenados uns dos outros. Dependendo das necessidades do sistema de indexação, *m* pode ser ajustado para agregar mais ou menos os dados indexados.

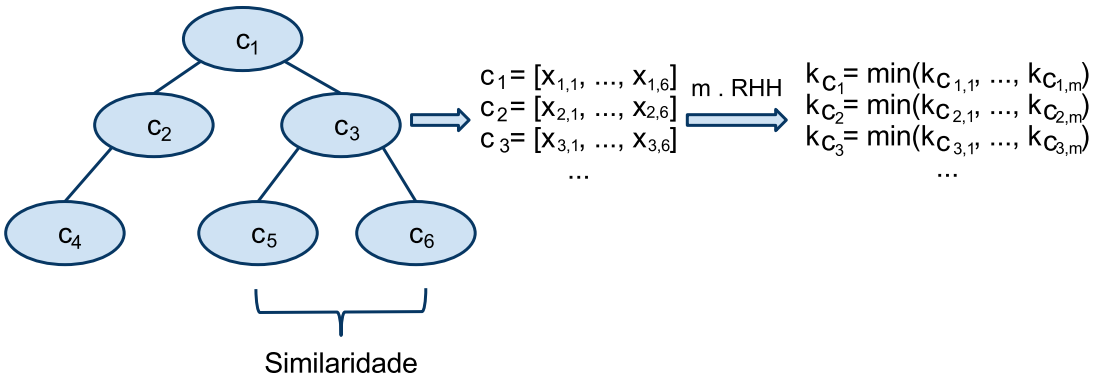


Fig. 6.9: Processo geral de criação dos identificadores.

É importante explicar a relação entre o fator *m* e a similaridade na criação dos identificadores. Se *m* for igual a 1, a similaridade entre cada par de vetores de conceito, medida pelo cosseno do ângulo entre eles, é preservada na similaridade de Hamming entre os identificadores. Por outro lado, se o espaço de indexação for Euclidiano, como o do cenário apresentado nesta subseção e comumente utilizado em DHTs, a utilização dessas chaves pode resultar na indexação de chaves similares longe umas das outras, como explicado na subseção anterior. Para tratar esse problema, são utilizados valores maiores para o fator *m*. Como visto, *m* grupos de vetores $\overrightarrow{r}$ são criados e a menor chave gerada é escolhida como chave daquele conceito. Fazendo dessa forma, há uma grande probabilidade de conceitos similares gerarem a menor chave de um mesmo grupo de vetores $\overrightarrow{r}$ entre os *m* grupos.

Mas, conceitos que não são similares possuem grandes chances de gerarem os resultados a partir de grupos diferentes de vetores $\overrightarrow{r}$. Isso significa que, para pares de identificadores de conceitos não similares, a similaridade de Hamming é perdida, pois esses podem ter sido criados por diferentes grupos de vetores $\overrightarrow{r}$.

Entretanto, como para o cenário apresentado nesta seção é necessária a agregação de chaves similares próximas umas das outras no espaço Euclidiano, não havendo a necessidade da manutenção da similaridade na distância de Hamming, o uso de valores maiores que 1 para o fator *m* é necessário. Para esses valores de *m*, os conceitos são armazenados próximos uns dos outros nessa métrica.

Uma característica dessa abordagem é que ela leva em conta somente a topologia da ontologia, fazendo com que duas ontologias que possuam a mesma topologia, mas que definam domínios totalmente diferentes, gerem chaves de conceito idênticas. A Figura 6.10 apresenta essa situação: a Ontologia 1 possui topologia diferente da Ontologia 2, mas grande parte dos seus conceitos são idênticos, enquanto a Ontologia 3 possui a mesma topologia da Ontologia 1, mas todos os seus conceitos são diferentes. Essa situação indica que é necessária uma estratégia na criação das chaves de forma que, as chaves de conceito de ontologias que possuam topologias diferentes, mas possuam conceitos similares sejam armazenadas próximas umas das outras e chaves de conceitos de ontologias com a mesma topologia, mas que definam conceitos diferentes, sejam armazenadas distantes umas das outras.

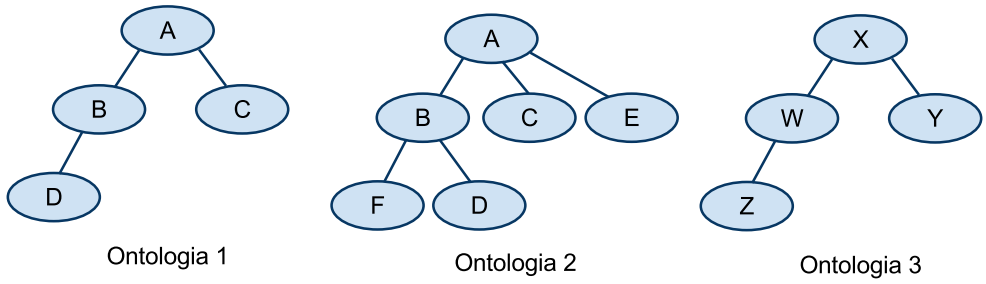


Fig. 6.10: Ontologias similares e dissimilares.

A estratégia adotada foi criar um prefixo para delimitar a indexação das chaves por ontologia. A próxima subseção apresenta como adicionar o vocabulário da ontologia na criação dos identificadores, de modo a evitar que se use somente a topologia das ontologias na determinação do índice de conceito.

6.2.2 Criação dos prefixos

O uso de prefixos nos identificadores dos conceitos distancia a indexação de ontologias que possuam a mesma topologia, mas que definem domínios diferentes. Ao mesmo tempo aproxima, no espaço virtual, duas ontologias com topologias diferentes que definem o mesmo domínio. Para essa

finalidade, é proposta a composição da chave de conceito em duas partes. A primeira é um prefixo gerado a partir do vocabulário da ontologia, também criado por uma função LSH. O sufixo é obtido como apresentado na Subseção 6.2.1.

O prefixo é criado utilizando a função LSH *Min-Wise* (Seção 4.2.1). Essa função é aplicada a um vetor de vocabulário, criado para cada ontologia, composto pelos nomes de todos os conceitos. Duas ontologias similares terão grandes chances de possuírem o mesmo prefixo, dada a similaridade entre seus vocabulários. É importante citar que nesta tese é considerado que ontologias de um mesmo domínio seguem um mesmo vocabulário, previamente definido, para nomear seus conceitos. A geração do prefixo é feita da seguinte maneira, também apresentada na Figura 6.11:

- Para cada ontologia, um conjunto de vocabulário *v* é criado, composto pelos nomes de todos os conceitos presentes na ontologia;
- *v* é convertido em um conjunto de inteiros *v'* por meio de uma função de *hash* tradicional (MD5, SHA-1, etc.) para cada elemento de *v*;
- o vetor *v'* é utilizado em uma função *Min-wise*, processo descrito na Seção 4.2.1;
- o resultado do Min-wise de *v'*, ou seja, o menor inteiro gerado após as permutações, é usado como prefixo em todos os identificadores de conceitos daquela ontologia.

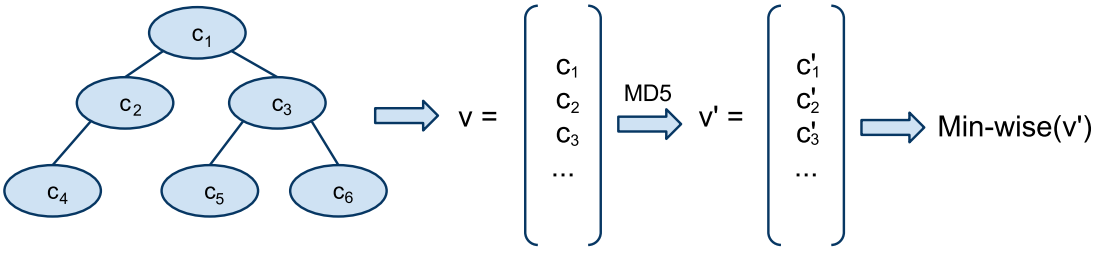


Fig. 6.11: Criação dos prefixos.

Dessa forma, ontologias similares, que definam o mesmo domínio, por exemplo *Música*, terão alta probabilidade de gerar o mesmo prefixo, pois provavelmente possuem vocabulários similares e serão armazenadas na mesma região do espaço virtual da DHT. Ontologias que definam outros domínios, por exemplo *Esporte*, serão armazenadas em outras regiões do espaço virtual, balanceando o espaço de endereçamento da DHT. É importante citar que a abordagem aqui utilizada parte do pressuposto que um vocabulário padrão é seguido para a definição de um domínio. A Figura 6.12 mostra uma possível distribuição de três domínios diferentes em uma rede P2P.

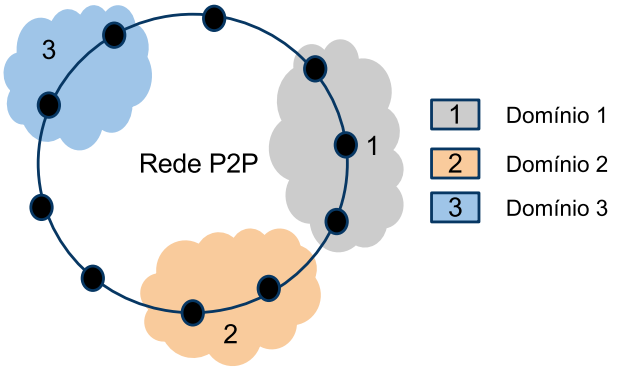


Fig. 6.12: Distribuição dos domínios na rede P2P.

Essa agregação de diferentes ontologias, que definam o mesmo domínio de conhecimento, em uma mesma região do espaço de indexação possibilita que uma busca por um conceito de uma ontologia possa retornar dados indexados em outra ontologia que também defina o mesmo domínio de conhecimento. Dentro de uma região pode existir uma gerência que identifica a ocorrência de um mesmo conceito em diversas ontologias e o resultado de uma busca baseada em uma dessas ontologias pode ser completada com dados classificados por outra. Por exemplo, na Figura 6.10, provavelmente uma busca por dados classificados no conceito *D* da Ontologia 1 pode ser completada por dados classificados no mesmo conceito da Ontologia 2, incluindo conceitos similares. Pelo fato das ontologias estarem indexadas na mesma região, a busca por esses dados é facilitada. A indexação dentro de uma região será discutida na Seção 6.6.

A próxima subseção apresenta alguns testes feitos e os resultados obtidos. Os testes visam mostrar o comportamento da agregação dos identificadores gerados pela função LSH no espaço virtual de uma DHT e os ganhos obtidos com essa abordagem.

6.3 Avaliação

A avaliação da metodologia de utilização da função LSH, apresentada nas seções anteriores, foi feita em duas partes. Na primeira foi avaliada a agregação das chaves em relação à variação do fator *m* e o tamanho do prefixo. Na segunda, foram feitos testes de busca em uma DHT, medindo-se o custo na busca por conceitos similares. A geração dos sufixos das chaves foi feita utilizando o método baseado no algoritmo do eTVSM, mas como visto nos resultados apresentados no Capítulo 3, qualquer outro método de similaridade entre conceitos de uma mesma ontologia poderia ter sido utilizado.

6.3.1 Avaliação para um cenário utilizando uma ontologia do domínio *Música*

Os testes feitos objetivam mostrar o impacto da agregação e os ganhos proporcionados pela função LSH atrelada a uma ontologia real, em um espaço de indexação de uma DHT baseada no *Chord*. Os testes utilizaram identificadores de 128 bits e uma ontologia baseada em alguns dos conceitos existentes no diretório de categorias do *Yahoo!* sob o conceito *Música*, mostrada na Figura 6.13. Essa classificação pode ser utilizada para a indexação de textos, vídeos, imagens sobre música ou até mesmo arquivos de áudio. Para ilustrar, a Tabela 6.3.1 apresenta a similaridade entre os conceitos da ontologia do domínio *Música* definida pelo cosseno do ângulo entre os vetores de cada conceito.

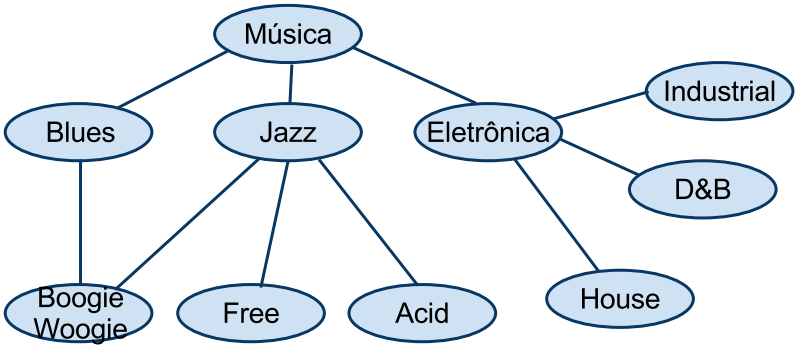


Fig. 6.13: Ontologia *lightweight* para o domínio *Música*.

Tab. 6.1: Tabela de similaridade do método *eTVSM* para os conceitos da ontologia do domínio *Música* da Figura 6.13.

Conceito	Música	Blues	Jazz	Elet.	Indust.	BW	Free	Acid	House	DB
Música	1,000	0,871	0,909	0,704	0,621	0,871	0,735	0,735	0,621	0,621
Blues	0,871	1,000	0,836	0,327	0,289	1,000	0,577	0,577	0,289	0,289
Jazz	0,909	0,836	1,000	0,420	0,371	0,836	0,871	0,871	0,371	0,371
Elet.	0,704	0,327	0,420	1,000	0,882	0,327	0,378	0,378	0,882	0,882
Indust.	0,621	0,289	0,371	0,882	1,000	0,289	0,333	0,333	0,667	0,667
BW	0,871	1,000	0,836	0,327	0,289	1,000	0,577	0,577	0,289	0,289
Free	0,735	0,577	0,871	0,378	0,333	0,577	1,000	0,667	0,333	0,333
Acid	0,735	0,577	0,871	0,378	0,333	0,577	0,667	1,000	0,333	0,333
House	0,621	0,289	0,371	0,882	0,667	0,289	0,333	0,333	1,000	0,667
DB	0,621	0,289	0,371	0,882	0,667	0,289	0,333	0,333	0,667	1,000

O primeiro teste consistiu em, variando-se o valor de m (Seção 6.2.1), constatar o seu impacto na agregação das chaves. A Figura 6.14 mostra, para m variando em 1, 5, 10, 20 e 50, a faixa de distribuição dos identificadores em um espaço virtual de 128 bits em forma de anel, dividido em 2000

partes iguais, cada parte representando um nó em uma DHT. A faixa é calculada pelo número de nós existentes entre o primeiro e o último nó que armazenam identificadores no espaço virtual. Nesse teste, o prefixo mostrado na Seção 6.2.2 não foi utilizado, já que o intuito é mostrar o impacto na agregação somente pela variação do valor de *m*.

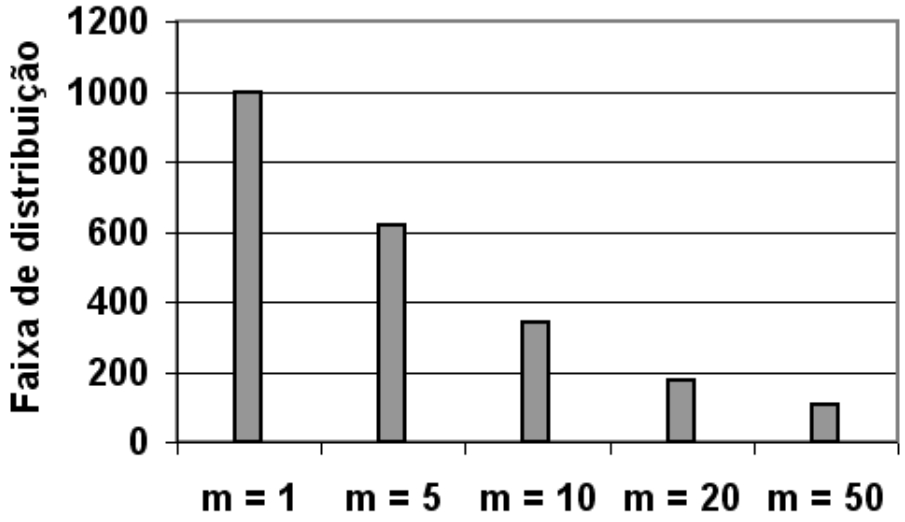


Fig. 6.14: Impacto do *m* na faixa de distribuição.

A Figura 6.14 mostra a média de 10 testes em cada coluna. Como pode ser visto, o resultado se assemelha ao obtido na Seção 6.3.2, ou seja, quanto maior o valor de *m*, menor é a faixa de distribuição dos identificadores. Isso mostra que *m* é um fator que pode ser ajustado de acordo com a necessidade do sistema, caso seja necessário uma maior ou menor agregação dos identificadores. Os intervalos de confiança de 95% são, para *m* com valores iguais a 1, 5, 10, 20 e 50, respectivamente: 180,48, 188,53, 95,51, 64,52 e 36,94.

A Figura 6.15 mostra a relação da agregação dos identificadores em relação ao número de nós existentes na DHT e a variação do tamanho do prefixo utilizado nos identificadores. Para esses testes, *m* foi fixado em 20 e o número de nós na DHT variou de 1000 a 100000.

É possível ver que, quanto maior o tamanho do prefixo, mais agregados os identificadores se encontram e, quanto maior o tamanho da DHT, mais nós compartilham os identificadores de uma mesma ontologia. Para comparação, o mesmo teste feito com uma função tradicional de *hash* (MD5) possui um crescimento linear, ocupando praticamente todo o espaço do anel. Por motivos de escala, esse resultado não foi colocado no gráfico.

Esse resultado mostra que o prefixo atua na agregação da seguinte forma: quanto mais bits são usados no prefixo, o espaço virtual da DHT é dividido em um número maior de partes para a indexação de diversas ontologias. Dessa forma, quanto maior o prefixo usado nas chaves de conceito de uma

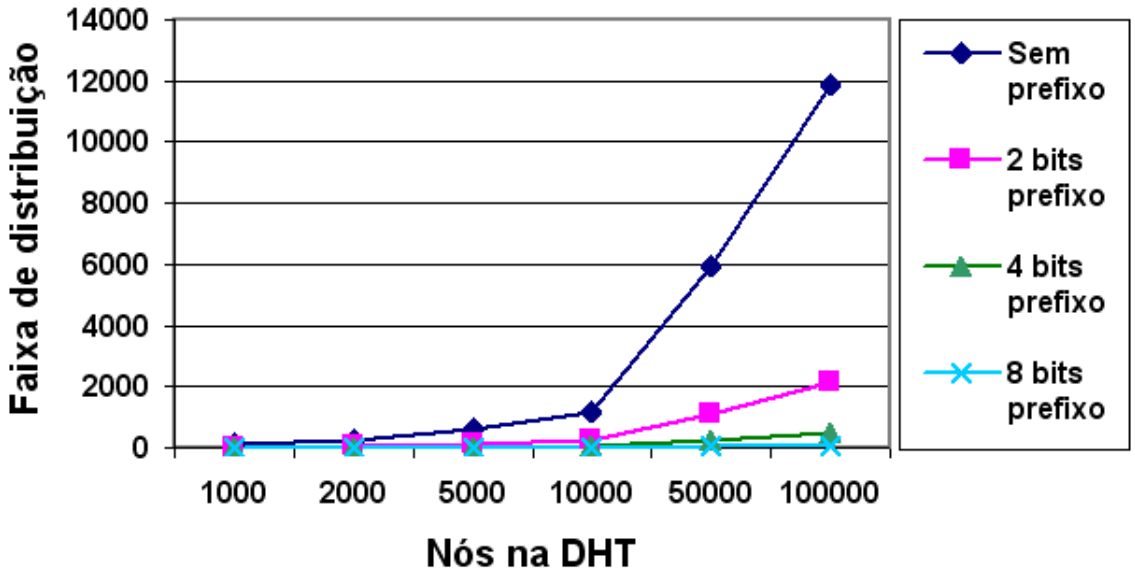


Fig. 6.15: Variação no tamanho da DHT.

ontologia, menor será a parte do espaço virtual na qual as chaves dessa ontologia estarão indexadas. O número de nós existentes em cada parte varia de acordo com a distribuição e a quantidade de nós existentes na DHT.

Deve ser lembrado que, se a princípio o resultado obtido com a função LSH pode sinalizar um desbalanceamento no espaço virtual da DHT, uma vez que a tendência seria os identificadores serem agregados no início do anel, devido ao fato da escolha do menor valor gerado como resultado do *hash*, o sistema proposto deve indexar diversas ontologias, definindo diversos domínios de conhecimento. Como mostrado na Figura 6.12, a inserção de várias ontologias no sistema, utilizando prefixo, resulta no preenchimento do anel pelas regiões que essas ocuparão.

A Figura 6.16 mostra a distância, na média, entre o identificador de um dos conceitos em relação a todos os identificadores dos outros conceitos, variando-se o tamanho do prefixo e mantendo o número de nós no espaço virtual em 2000 e o valor de m igual a 20. Nesse teste, o conceito escolhido foi o *Free*. Esses valores são comparados também com o mesmo teste feito com o MD5.

Novamente, é possível ver que, quanto maior o tamanho do prefixo utilizado, mais próximo os identificadores dos conceitos são armazenados. A média para prefixos com 8 bits, ou mais, torna-se praticamente 0, o que significa que, para este espaço virtual dividido em 2000 partes, praticamente todos os conceitos foram armazenados no mesmo nó da DHT. Para os identificadores gerados pela função de *hash* tradicional, a distância entre os conceitos é maior, o que resulta em um maior balanceamento da DHT, porém, em um maior custo na busca por conceitos relacionados, como mostrado no próximo teste. Na Figura 6.16, os intervalos de confiança de 95% para os testes “Sem Prefixo”, “2

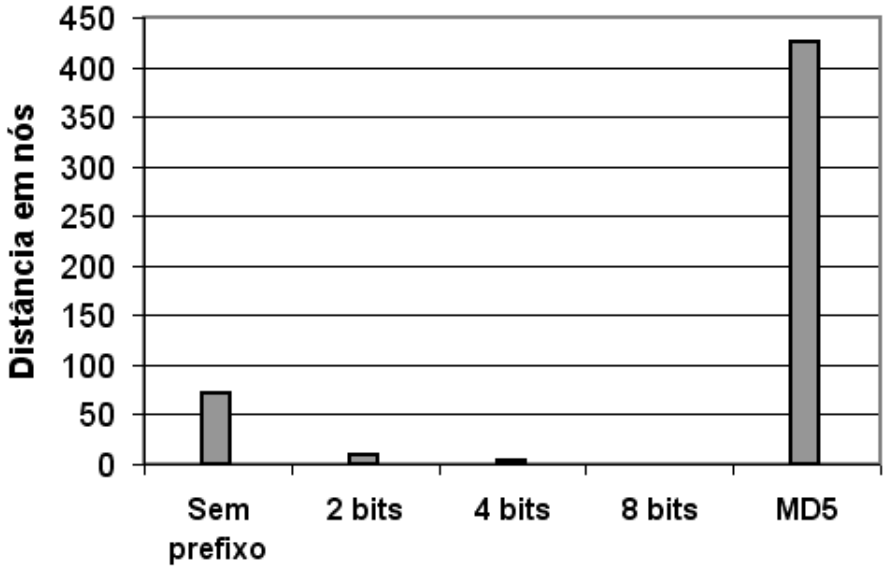


Fig. 6.16: Distância do conceito *Free* aos demais conceitos da ontologia.

bits", "4 bits", "8 bits" e "MD5" são, respectivamente, 6,09, 2,41, 0,3, 295,94.

A Figura 6.17 mostra o número de saltos necessários para, a partir do nó que armazena o identificador de um conceito, atingir todos os outros conceitos da ontologia. Como no teste anterior, o termo escolhido foi o *Free* e foi variado o tamanho do prefixo. Nesse teste foi empregado o simulador de rede P2P *Oversim*[4], utilizando 2000 nós no anel da DHT. Esse teste foi feito em duas DHTs diferentes: no *Chord* e em uma baseada no fenômeno do *Small World* [87]. A diferença entre as duas DHTs se refere à criação dos *fingers* entre os nós. No *Chord*, o estabelecimento dos *fingers* é feito de forma que não há preferência para sua criação, enquanto que a DHT baseada no fenômeno do *Small World* privilegia a criação de *fingers* entre nós que estejam próximos no espaço virtual [87], como apresentado no início da Seção 6.4.

Devido ao roteamento do *Chord* ser feito no sentido horário, para uma maior justiça nos testes, as chaves foram ordenadas e buscadas na ordem crescente. Ou seja, para uma chave de conceito C_k qualquer, se o valor da chave conceitual de *Free* é menor que C_k, os saltos necessários a partir de *Free*, com destino à C_k, foram contabilizados. Caso contrário, contabilizaram-se os saltos a partir de C_k com destino à chave do conceito *Free*.

Como pode ser visto na Figura 6.17, em ambas DHTs, o número de saltos necessários para atingir todos os identificadores de conceitos a partir do identificador do conceito *Free* é menor se comparado com número de saltos necessários para o mesmo teste, mas considerando identificadores criados pelo MD5. Os intervalos de confiança de 95% são, para o *Chord* "Sem Prefixo", "2 bits de Prefixo", "4

[4]Oversim - http://www.oversim.org (acessado em 18 de maio de 2011).

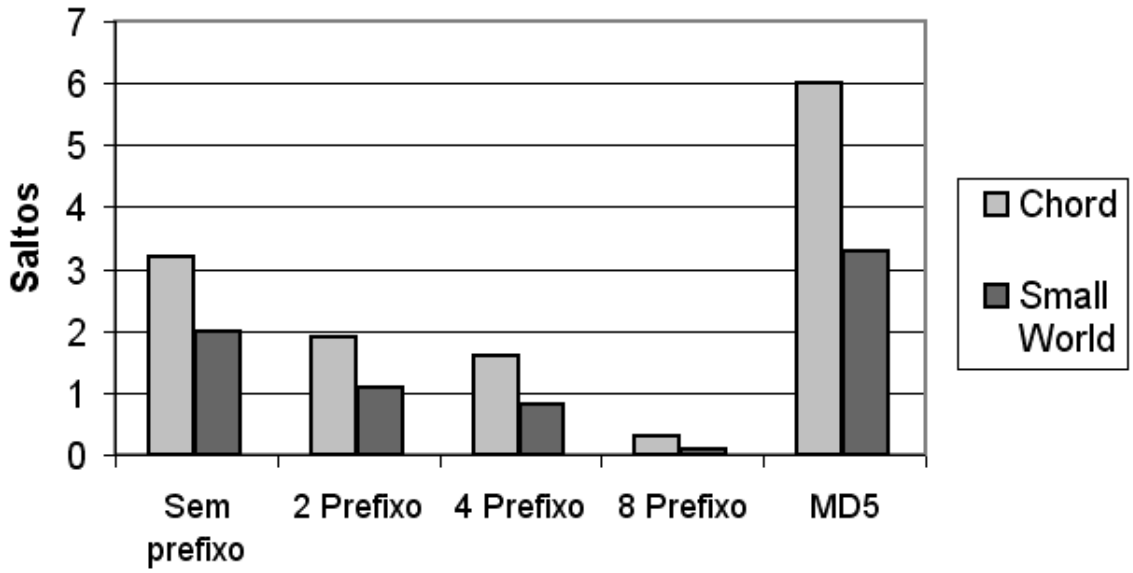


Fig. 6.17: Saltos em relação a *Free*.

bits de Prefixo", "8 bits de Prefixo" e "MD5", respectivamente: 1,77, 1,03, 0,89, 0,59 e 0,86. Para os testes utilizando a DHT *Small World* com "Sem Prefixo", "2 bits de Prefixo", "4 bits de Prefixo", "8 bits de Prefixo" e "MD5", os intervalos de confiança de 95%, respectivamente, são: 0,8, 0,7, 0,44, 0,2 e 0,93.

6.3.2 Avaliação para um cenário utilizando três ontologias

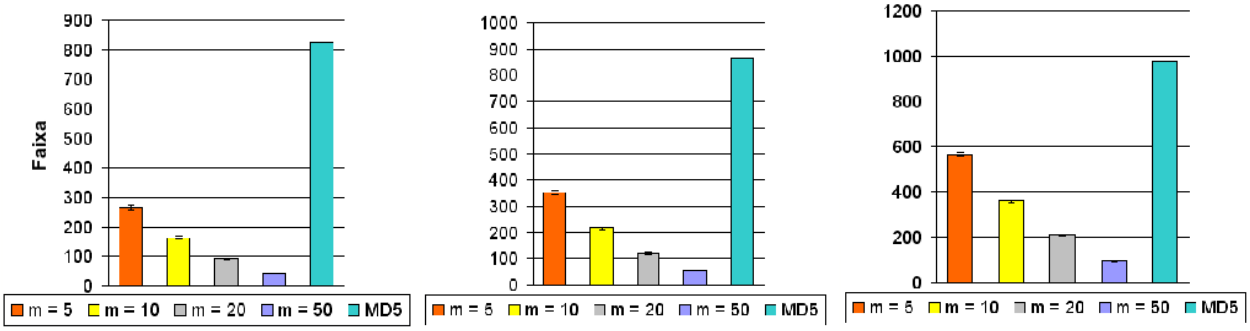
O teste apresentado nesta subseção mostra a agregação dos conceitos de ontologias em um anel *Chord*. Um espaço de 128 bits foi dividido em 1000 recipientes de tamanhos iguais e foram criadas chaves conceituais de 128 bits. Cada recipiente pode conter até 2^{128}/1000 chaves. As chaves foram distribuídas entre os recipientes, sendo a decisão sobre qual recipiente recebe quais chaves baseada no valor de cada chave. Nos gráficos é mostrada a faixa de distribuição das chaves no espaço de indexação, medida pela diferença entre o maior e o menor recipiente que receberam chaves, considerando o sentido horário do anel. Quanto menor a faixa, mais próximos as chaves conceituais estão uma das outras no espaço virtual. Alguns dos testes feitos são os mesmos apresentados na Seção 6.3.1, mas agora utilizando três ontologias. Novamente foi utilizado o método baseado no algoritmo do eTVSM, mas como visto nos resultados apresentados do Capítulo 3, qualquer outro método de similaridade entre conceitos de uma mesma ontologia poderia ter sido utilizado.

Nesse teste foram usadas três diferentes ontologias: ontologia 1 é a mesma da Figura 3.3 e as ontologias 2 e 3 são árvores binárias com 15 (c_1 a c_{15}) e 127 conceitos (c_1 a c_{127}), respectivamente. Os testes foram feitos variando-se o fator *m* (número de chaves criadas), esse assumindo os valores

5, 10, 20 e 50. Cada teste foi feito 1000 vezes, e os gráficos mostram a média dos resultados com intervalo de confiança de 95%. Os mesmos testes foram feitos usando a função de hash MD5 para comparação.

Como pode ser visto na Figura 6.18, quanto mais chaves são criadas para as três ontologias, em outras palavras, quanto maior o valor do fator *m*, mais agregadas as chaves são armazenadas. Mesmo para o menor valor de *m* testado, os resultados são melhores do que os que utilizam a função de hash tradicional. Para valores maiores de *m*, a probabilidade de que vetores de conceitos similares gerem valores similares aumenta, assim como a sua agregação.

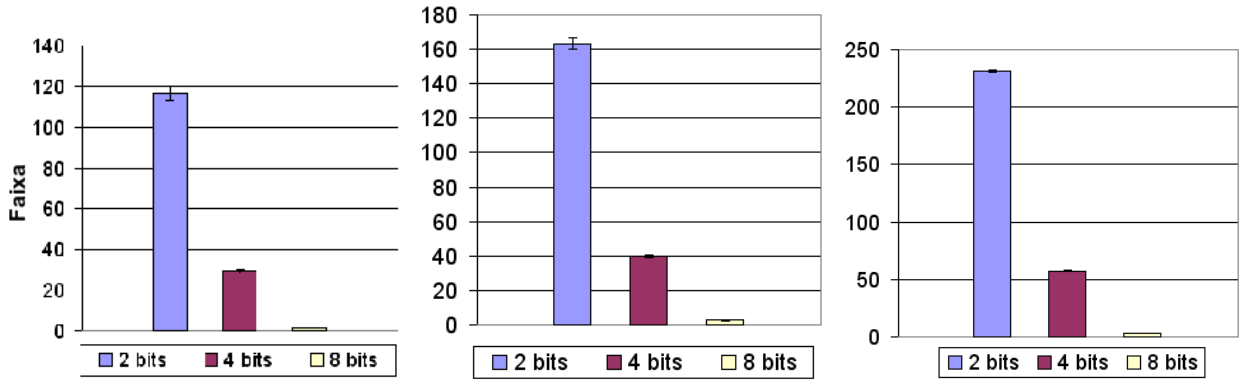
Esses resultados mostram que efetivamente *m* é um fator de agregação. Dependendo da aplicação, ou dos requisitos do sistema, um valor maior ou menor de *m* pode ser utilizado.



(a) Ontologia 1 (b) Ontologia 2 (c) Ontologia 3

Fig. 6.18: Faixa de distribuição - sem o uso de prefixo.

A Figura 6.19 apresenta os resultados para os testes usando as mesmas três ontologias, mas com prefixo de 2, 4 e 8 bits, como apresentado na Subseção 6.2.2, e o fator *m* foi fixado em 1. Os testes mostram o impacto do prefixo na agregação das chaves quando não há variação no valor de *m*.



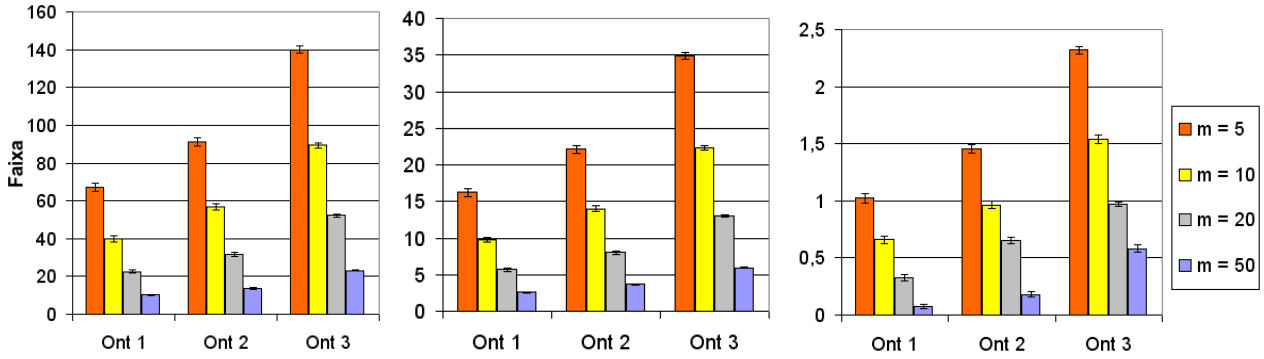
(a) Ontologia 1 (b) Ontologia 2 (c) Ontologia 3

Fig. 6.19: Faixa de distribuição - usando prefixo e *m* = 1.

Os resultados da Figura 6.19 mostram que quanto maior o prefixo, mais agregadas as chaves são armazenadas. A razão para isso é que prefixos maiores dividem o espaço de indexação em mais partes, como visto na Seção 6.2. Portanto, quanto maior o prefixo, menor será a parte do espaço virtual no qual as chaves conceituais de um ontologia serão armazenadas.

Os testes seguintes usam as mesmas ontologias previamente apresentadas e também utilizam prefixos nas chaves. Os tamanhos dos prefixos testados foram 2, 4 e 8 bits. A segunda parte das chaves foi gerada usando a mesma variação para *m* como na Figura 6.18.

Nas Figuras 6.20(a), 6.20(b) e 6.20(c), os resultados mostram que ao se utilizar um prefixo, as chaves tendem a ser armazenadas próximas, na média. Quanto menos bits são usados no prefixo, mais os resultados se aproximam dos obtidos sem o uso do prefixo. Devido à escala adotada, os intervalos de confiança de 95% nos gráficos quase não são notados nas figuras. Considerando um espaço de 128 bits dividido em 1000 recipientes, chaves conceituais de ontologias similares com 10 bits de prefixo ou mais, são armazenadas no mesmo recipiente. É possível concluir que, para um cenário de 1000 recipientes, 4 e 8 bits são boas escolhas para o tamanho do prefixo.



(a) Prefixo de 2 bits (b) Prefixo de 4 bits (c) Prefixo de 8 bits

Fig. 6.20: Faixa de distribuição - uso de prefixo.

Distância entre conceitos

Neste teste foram utilizadas três ontologias e foi utilizado prefixo nas chaves. A ontologia 1 é a mesma da Figura 3.3, ontologia 2 é composta de 10 conceitos (conceitos de c_1 a c_7 são os mesmo da ontologia 1 e mais três: c_8 a c_{10}) e a ontologia 3 possui a mesma topologia da ontologia 1, entretanto possui conceitos totalmente diferentes (c_{16} a c_{22}). As ontologias podem ser vistas na Figura 6.21.

O objetivo deste teste é mostrar a distância entre: (i) dois conceitos similares pertencentes a duas ontologias diferentes e; (ii) dois conceitos totalmente diferentes pertecentes a duas ontologias com a mesma topologia. Para testar (i), os conceitos escolhidos para medir a distância foram c_5 da ontologia 1 e o mesmo na ontologia 2. Apesar deles serem o mesmo conceito, ambos são localizados

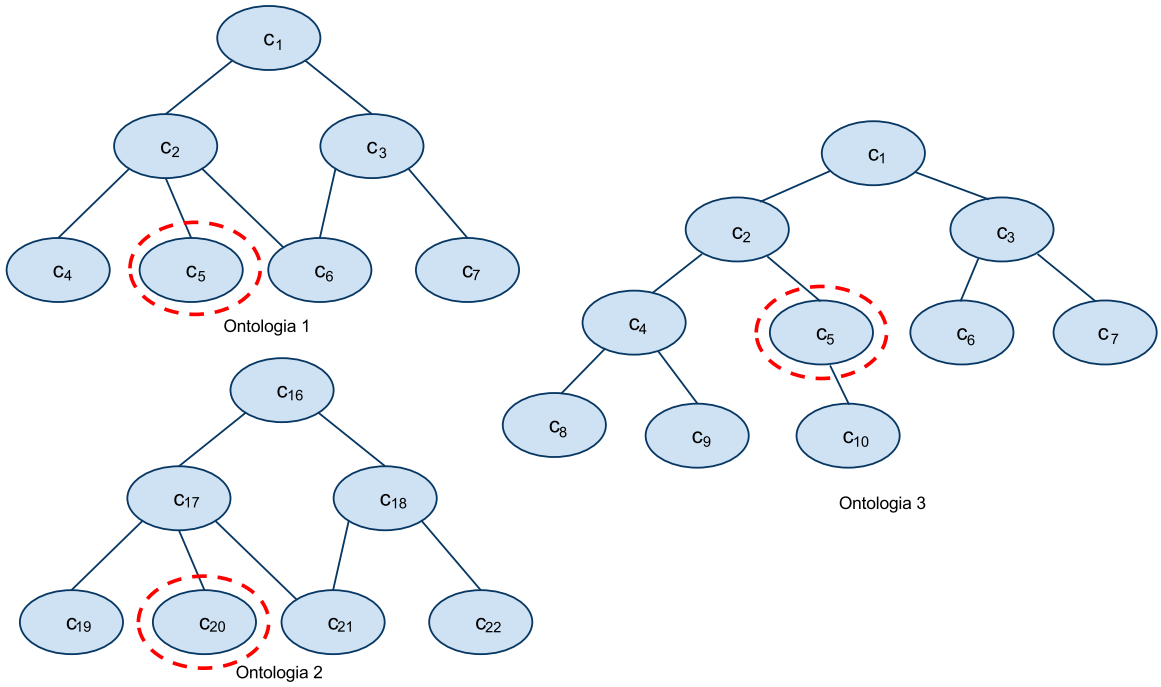
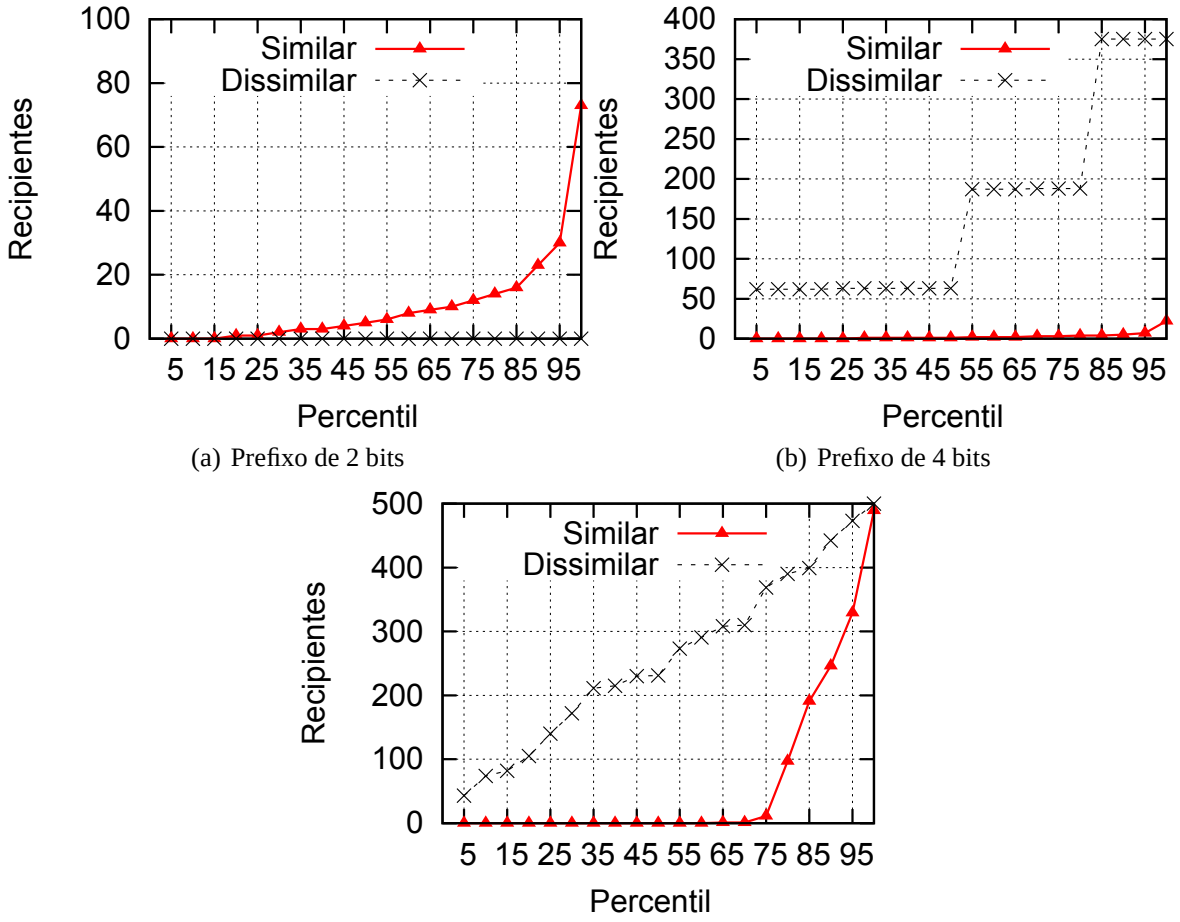


Fig. 6.21: Ontologias utilizadas nos testes.

em posições diferentes em suas ontologias correspondentes. Para testar (ii), os conceitos escolhidos foram c_5 da ontologia 1 e c_{20} da ontologia 3, devido ao fato de ambos estarem localizados no mesmo ponto das ontologias. Isso significa que seus vetores de conceito são iguais. A ideia é constatar o impacto do prefixo nas chaves geradas. Os três conceitos estão marcados com círculos na Figura 6.21.

A Figura 6.22 mostra os percentils para 1000 testes. A linha "Similar" corresponde à distância, medida em quantidade de recipientes, entre c_5 da ontologia 1 e o mesmo conceito da ontologia 2, e a linha "Não similar" corresponde à distância de c_5 da ontologia 1 e c_{20} da ontologia 3. O fator *m* foi fixado em 20, mas resultados similares foram obtidos com outros valores.

Apesar da Figura 6.22(a), é possivel ver que na maior parte dos testes, conceitos similares estão próximos uns dos outros e conceitos que não são similares estão distantes uns dos outros. A razão para que os testes mostrados na Figura 6.22(a) não serem bem sucedidos é que, para prefixo com 2 bits, há grande probabilidade que mesmo duas ontologias não similares gerem o mesmo menor valor para ser escolhido como prefixo, pois as opções são poucas (2^2 combinações). Esse fato resulta em toda carga semântica ser localizada na segunda parte da chave. Como a segunda parte é criada considerando somente a topologia das ontologias, as chaves de dois conceitos na mesma posição topológica em suas respectivas ontologias (ambas possuindo a mesma topologia mas conceitos diferentes) são armazenadas no mesmo recipiente. Entretanto, ao se utilizar prefixos com 4 e 8 bits (Figuras 6.22(b)



(a) Prefixo de 2 bits

(b) Prefixo de 4 bits

(c) Prefixo de 8 bits

Fig. 6.22: Distância entre termos usando m = 20.

e 6.22(c)), na maioria dos testes, o conceito c_5 da ontologia 1 estava localizado perto do conceito c_5 da ontologia 2 (por exemplo, para dois conceitos similares, usando 4 bits de prefixo, pelo menos 90% dos testes resultaram em uma distância de 0 entre eles) e distante do conceito c_{20} da ontologia 3. Isso confirma que não somente a topologia da ontologia está sendo considerada na criação das chaves, mas também o seu vocabulário.

Saltos em uma DHT

Este teste mostra como o uso de funções LSH, para a criação de identificadores com prefixo, pode melhorar a busca conceitual em uma rede P2P. Usando o simulador *Oversim*, foi criado um anel *Chord* formado por 1000 nós. Neste teste foram usadas as ontologias similares da Figura 6.21 (ontologias 1 e 2). Foram feitos quatro testes: um utilizando a função de *hash* MD5 e outros três variando em 2, 4 e 8 bits o tamanho do prefixo utilizado nas chaves. Em cada teste, as chaves de conceitos das duas

ontologias foram indexadas no mesmo anel. O fator *m* foi fixado em 20. Em cada teste foi computada a média do número de saltos necessários para atingir todas as chaves indexadas no anel, partindo do nó que armazena c_5 da ontologia 1. Devido ao roteamento do *Chord* ser no sentido horário, em todos os testes as chaves foram ordenadas em ordem crescente e procuradas nessa ordem. Ou seja, se a chave de conceito de c_5 é menor que a chave de conceito de c_4, os saltos foram contabilizados do nó que armazena c_5 até o nó que armazena c_4, caso contrário, o oposto foi contabilizado. A Tabela 6.3.2 mostra os resultados.

Tab. 6.2: Média de saltos em uma DHT.

-	2 bits	4 bits	8 bits	MD5 (sem prefixo)
Saltos	3.12	2.35	0	5.06

Como mostrado na Tabela 6.3.2, quanto mais agregadas estiverem as chaves no espaço de indexação, menos saltos são necessários para recuperar qualquer outra chave, partindo de uma chave arbitrária. Para 8 bits ou mais de prefixo, quase todas as chaves foram armazenadas no mesmo nó, resultando em uma média de aproximadamente 0 saltos. A agregação faz uma busca conceitual recursiva ser menos custosa se comparada com o resultado obtido utilizando qualquer função de *hash* tradicional, como o MD5.

6.4 Cenário que utiliza a distância de Hamming

Nessa variação do cenário proposto foi utilizada uma DHT estruturada baseada no fenômeno *Small World* [88][87] e que utiliza a distância de Hamming como métrica. Na literatura encontra-se que o fenômeno *Small World*, originalmente estudado em [89], se mostra como uma maneira interessante de agregar dados similares em uma DHT, mantendo um custo baixo para a busca de conteúdos similares em relação ao número de saltos necessários para a obtenção dos dados relacionados [90]. Esse fenômeno define que nós da rede que estão situados próximos tenham uma alta taxa de interligação, ou seja, alta probabilidade de existir um *finger* entre eles, e baixa probabilidade de existir *fingers* distantes, ou seja, alguns poucos *fingers* interligando nós distantes. Essa formação reduz o custo, em termos de saltos para, a partir de um nó, atingir um outro qualquer que esteja próximo e mantém baixo o número de saltos para se atingir um nó distante.

Nessa DHT, o espaço de endereçamento é ordenado segundo o código de Gray e dividido em regiões. As regiões são representadas por um identificador de *n* bits sendo que, para qualquer região, a diferença entre o identificador dessa região e o identificador das suas regiões sucessoras e predecessoras é de 1 bit, como definido pela sequência do código de Gray. Entre as regiões são criados

fingers, levando em conta o fenômeno *Small World* e a distância de Hamming entre os identificadores de regiões, para determinação de proximidade. Em outras palavras, a criação de *fingers* entre nós que se encontram próximos na distância de Hamming é mais provável do que com aqueles mais distantes na mesma métrica. Como as chaves identificadores de conceitos apresentadas nesta tese mantêm conceitos similares próximos na distância de Hamming, a organização da DHT baseada no fenômeno *Small World* se torna adequada. A Figura 6.23 mostra esse cenário.

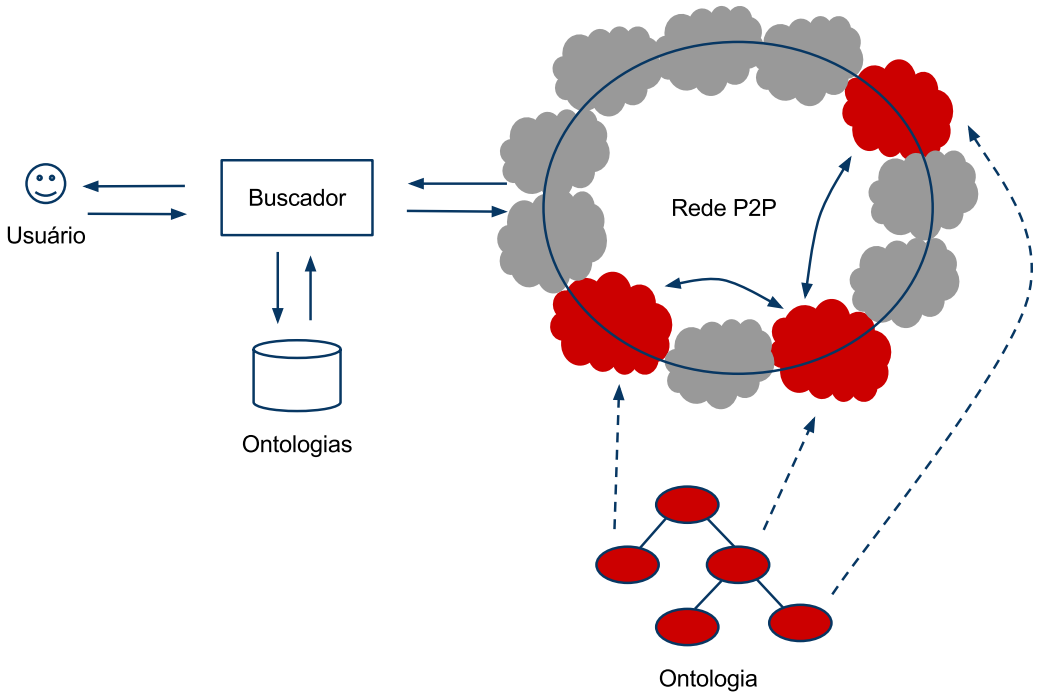


Fig. 6.23: Organização do sistema.

A função *hash* apresentada nesta tese foi utilizada para a criação de chaves de conceito que coincidem com os identificadores de regiões. Ou seja, cada região acomoda um conceito de uma ontologia que são interligados considerando a distância de Hamming. Dessa forma, cada região que representa um conceito possui grande probabilidade de possuir uma interligação com outras regiões que representam conceitos similares. Isso acontece devido ao fato de que chaves de conceitos similares, geradas pela função LSH, possuem baixa distância de Hamming entre si.

É importante notar que o identificador de cada região pode ser usado como prefixo para as chaves armazenadas naquela região. Dessa forma, dentro de cada região pode existir uma outra forma de indexação, agora no nível de conteúdo. As regiões delimitam os dados indexados conceitualmente, e dentro de cada região a forma de organização é livre. Várias ontologias podem ser indexadas, utilizando diferentes conjuntos de vetores $\overrightarrow{r}$ na função LSH, um para cada ontologia, distribuindo os conceitos nas regiões no espaço de indexação. Apesar da ideia inicial desse cenário ter sido concebida

para buscas por similaridade entre conceitos de uma mesma ontologia, uma evolução poderia ser feita a partir de uma análise do vocabulário das ontologias, indicando conceitos similares em diferentes ontologias, criando alguns *fingers* que os unam. Isso permitiria uma busca por dados similares sem restringir o escopo em uma só ontologia. A Figura 6.24 ilustra essa indexação. A próxima seção apresenta os testes feitos para avaliar esse cenário.

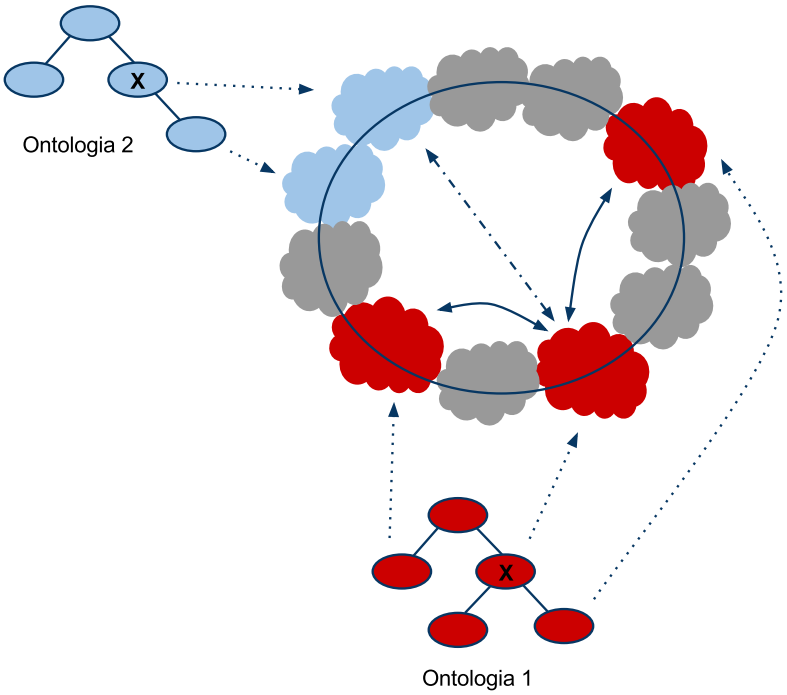


Fig. 6.24: Indexação de várias ontologias.

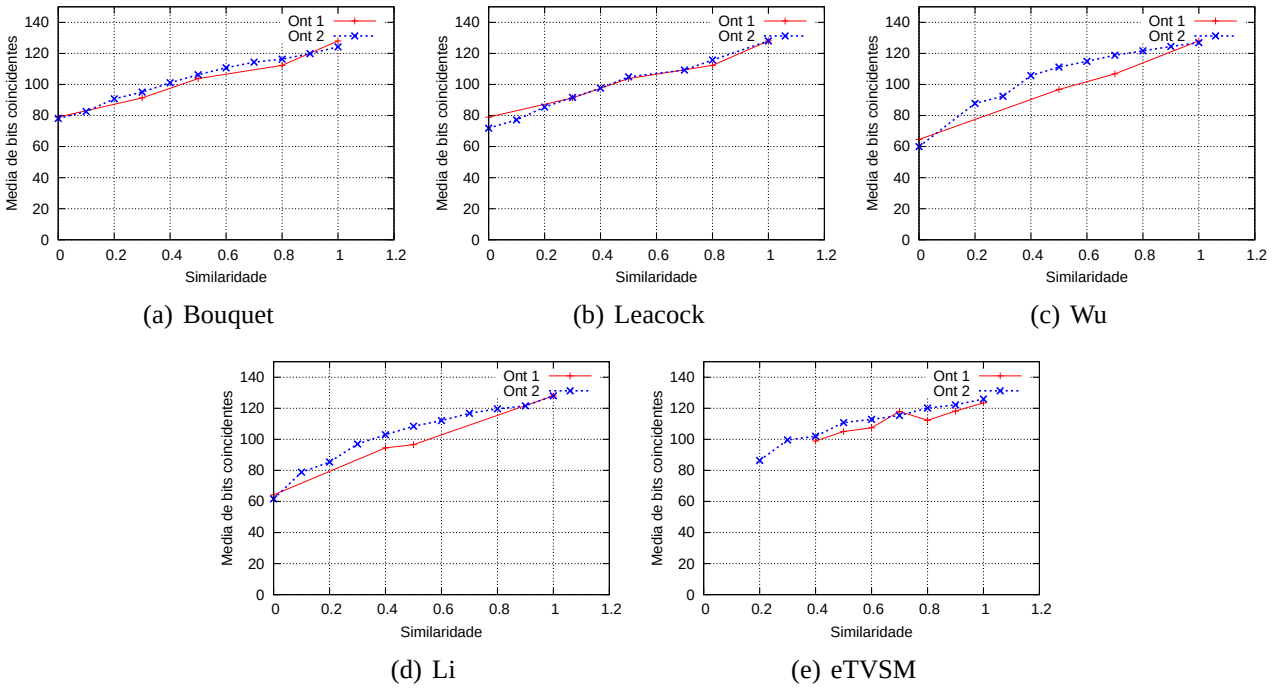
6.5 Avaliação

Com o objetivo de confirmar que conceitos similares possuem baixa distância de Hamming, foram feitos testes de avaliação dessa distância em relação aos métodos de similaridade entre conceitos de uma ontologia, apresentados no Capítulo 3. Nos testes, apresentados nas próximas subseções, foram utilizadas duas ontologias: ontologia 1, com 7 conceitos, apresentada na Figura 3.3 e ontologia 2, uma árvore binária com 127 conceitos. As chaves de conceito criadas possuem 128 bits.

6.5.1 Avaliação da distância de Hamming em relação à similaridade

O teste apresentado a seguir mostra a relação entre a distância de Hamming dos conceitos e a similaridade entre eles, definida pelos métodos de similaridade. Em outras palavras, foi calculada

para um nível de similaridade, por exemplo 0,5, a média da distância de Hamming medida entre as chaves de cada conceito e as chaves de todos os outros conceitos que possuem similaridade entre (0,4, 0,5] com o primeiro conceito.



(a) Bouquet (b) Leacock (c) Wu

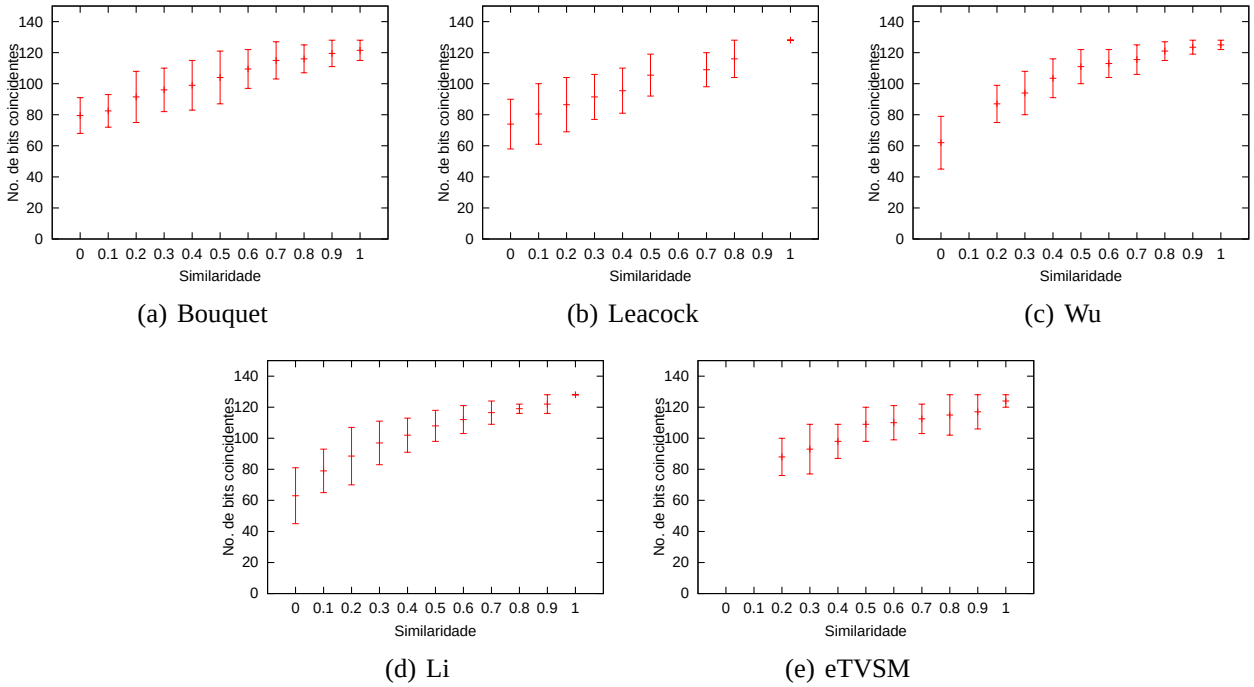
(d) Li (e) eTVSM

Fig. 6.25: Dispersão no espaço de Hamming pela similaridade (média, para todos os conceitos, da similaridade x número de bits coincidentes).

Os resultados são apresentados na Figura 6.25. O comportamento esperado é que, conforme a similaridade aumenta, o número de bits coincidentes também aumenta. Todos os métodos mostrados na figura possuem esse comportamento, o que mostra que a função de *hash* traduz a similaridade entre os conceitos para o valor resultante do hash, em termos de bits coincidentes. A diferença entre o teste utilizando a ontologia 1 e a ontologia 2 é a quantidade de níveis de similaridade. Para ontologias maiores, há mais níveis de similaridade para computar (mais pontos nas curvas da figura) devido ao fato de existirem mais conceitos.

O teste seguinte apresenta, para todos os conceitos da ontologia, a relação do número de bits coincidentes com a similaridade entre os conceitos, similaridade dada pelos métodos. Para cada nível de similaridade, foi calculada o menor e o maior número de bits coincidentes. A Figura 6.26 apresenta essas variações. Esse teste foi feito somente com a maior ontologia, devido ao fato de que a ontologia menor possui poucos níveis de similaridade.

Na Figura 6.26, a parte de baixo das barras representa o menor número de bits coincidentes comparando-se todos os conceitos que possuem aquele nível de similaridade entre si. O topo das



(a) Bouquet (b) Leacock (c) Wu

(d) Li (e) eTVSM

Fig. 6.26: Variação dos bits por nível de similaridade.

barras apresenta o número máximo de bits que coincidem para aquele nível de similaridade. Por exemplo, para o método *Leacock* (Figura 6.26(b)), para (0,4, 0,5] de similaridade, o número de bits coincidentes ficou entre 90 e 120 bits. Alguns gráficos não apresentam barra em determinados níveis de similaridade, o que significa que, para aquele método, não há pares de conceitos que possuam tal nível de similaridade para a ontologia utilizada.

É possível notar que para a ontologia utilizada, de 127 conceitos, o intervalo dos bits coincidentes é bem distribuído de acordo com o nível de similaridade. O comportamento de todos os métodos é como o esperado, ou seja, conforme a similaridade cresce, o número de bits coincidentes também cresce e a variação do intervalo tende a ser menor. Isso pode ser explicado pela alta probabilidade de dois conceitos bem similares terem bits coincidindo, portanto, a variação dos bits que não coincidem é menor e a quantidade de bits coincidentes é alta. Os métodos *Bouquet* e *Leacock* possuem intervalos maiores de bits coincidentes para cada nível de similaridade, pois esses métodos consideram mais conceitos similares entre si. Os outros métodos, *Wu* e *Li*, possuem intervalos menores, pois esses métodos possuem maior variação no nível de similaridade entre conceitos, como visto nas Figuras 5.7 e 5.9.

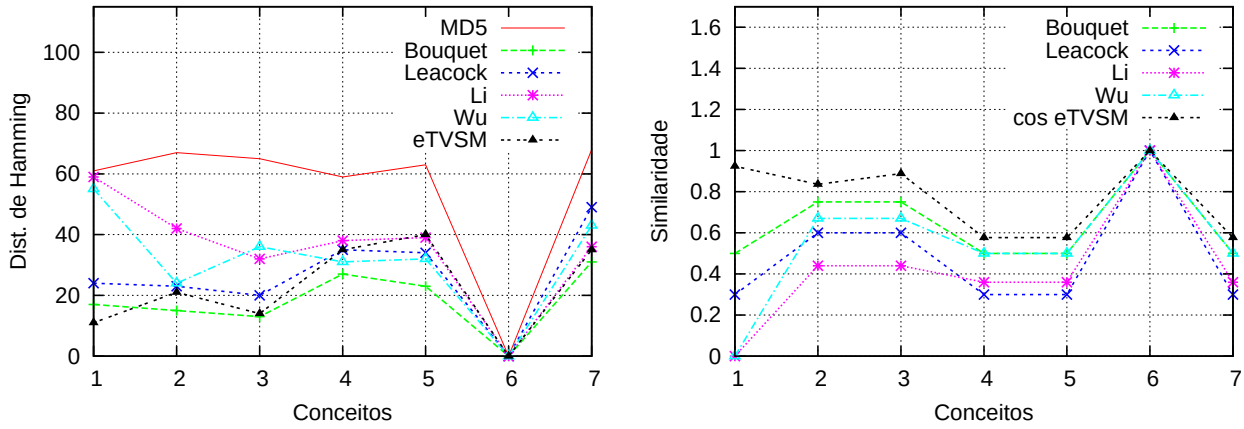
É possível ver que quanto maior a ontologia, melhor distribuída é a faixa de bits coincidentes de acordo com o nível de similaridade. O comportamento de todos os métodos é como o esperado, ou

seja, conforme a similaridade aumenta, o número de bits coincidentes também aumenta e a variação tende a ser menor. Isso pode ser explicado pela alta probabilidade de dois conceitos bastante similares terem bits coincidentes, portanto a variação dos bits que não coincidem é menor.

A próxima subseção apresenta a distância de Hamming entre conceitos específicos.

6.5.2 Distância de Hamming entre conceitos

Este teste mostra a relação entre a similaridade dada pelos métodos a as distâncias de Hamming correspondentes para um conceito específico. O objetivo é mostrar como as chaves estão espalhadas em um espaço hipercúbico que segue a distância de Hamming para distribuir dados. Um conceito foi escolhido para cada ontologia e é mostrada a similaridade daquele conceito com todos os outros. É também mostrada a distância de Hamming da chave daquele conceito com todas as outras chaves dos outros conceitos. A ideia é mostrar que a distância de Hamming segue a mesma curva obtida com os métodos. O conceito c_6 da ontologia 1 (Figura 3.3) e o conceito c_{127} (o último conceito folha, da esquerda para a direita) da ontologia 2 foram escolhidos. As Figuras 6.27 e 6.28 mostram os resultados. Os valores para a distância de Hamming obtidos usando a função MD5 também foram apresentados, para comparação.

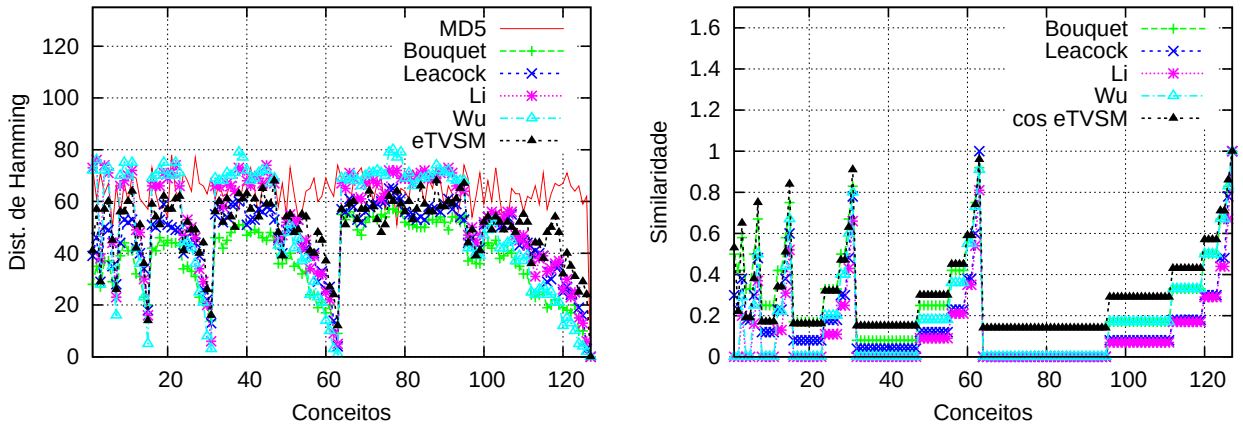


(a) Distância de Hamming

(b) Similaridade

Fig. 6.27: Ontologia 1 - 6.27(a) Distância de Hamming entre o conceito c_6 e todos os outros e 6.27(b) o nível de similaridade do conceito c_6 para todos os outros.

O resultado mostra que, para as duas ontologias, a curva de similaridade e a curva da distância de Hamming são inversamente correlacionadas. Em outras palavras, os conceitos que possuem pouca similaridade possuem maior distância de Hamming entre si e os conceitos mais similares estão próximos uns dos outros no espaço de Hamming. Esses resultados indicam que a similaridade



(a) Distância de Hamming (b) Similaridade

Fig. 6.28: Ontologia 2 - 6.28(a) Distância de Hamming entre o conceito c_{127} e todos os outros e 6.28(b) o nível de similaridade do conceito c_{127} para todos os outros.

entre conceitos é mantida no valor resultante do *hash* e, em um sistema de indexação hipercúbico, os elementos que são similares entre si serão indexados próximos uns dos outros. Como esperado, os resultados da distância de Hamming usando MD5 possuem um comportamento próximo à média (50% dos bits), como visto nas Figuras 6.27(a) e 6.28(a).

O teste mostrado a seguir consistiu em, a partir de cada região da DHT, representando um conceito, contabilizar a média dos saltos necessários para se atingir todos os outros conceitos. Nesse teste foi utilizada a ontologia de 127 conceitos e foram também feitos testes na DHT *Chord*, e na DHT baseada no fenômeno *Small World*, para comparação. As chaves de conceitos criadas tinham 20 bits, resultando em 2^{20} regiões. O método utilizado foi o baseado no eTVSM, mas poderia ter sido usado qualquer um dos outros métodos apresentados anteriormente no Capítulo 3.

Tab. 6.3: Média do número de saltos de acordo com a distância de Hamming das chaves de conceito.

Distância de Hamming	*Chord* + LSH	*Small World* + LSH	Redução
≤ 2	6.2	1.1	17,74%
≤ 4	8.1	2.7	33,33%
≤ 6	9.1	4.2	46,15%
≤ 8	9.6	5.4	56,25%
≤ 10	9.8	6.0	61,22%
> 10	9.9	6.3	63,63%

Os resultados apresentados na Tabela 6.3 mostram que o uso de uma DHT *Small World* reduz a média no número de saltos necessários para recuperar dados similares. O máximo de saltos obtidos[5] utilizando a DHT *Small World* foi de 11 e utilizando a DHT *Chord* foi obtido 20, uma redução de 45%. O melhor resultado obtido pela DHT que segue o fenômeno *Small World* mostra que o fato das chaves similares possuírem uma menor distância de Hamming entre si pode ser explorado por estruturas que levem isso em consideração, facilitando buscas por conceitos similares. Esse teste mostrou a aplicabilidade das chaves conceituais em estruturas de indexação distribuídas, facilitando a busca conceitual.

6.6 Discussão sobre a indexação dentro das regiões

Como visto nos cenários apresentados nas seções anteriores, o espaço de indexação da rede P2P é dividido em regiões, que abrigam as chaves que representam dados relacionados conceitualmente, indicando onde esses dados estão indexados de fato e como recuperá-los. A maneira como os dados propriamente ditos são indexados fica a critério de seus responsáveis.

A indexação desses dados pode ser feita em uma segunda DHT ou mesmo em um banco de dados de um repositório, como visto na Figura 6.29. Dentro dessas estruturas de indexação pode existir uma segunda forma de organização dos dados, baseada em outro critério de comparação entre eles como, por exemplo, seu conteúdo. Como visto no Capítulo 1, a literatura possui vários trabalhos que comparam dados de acordo com seu conteúdo, o que pode ser aproveitado em um sistema como proposto nesta tese. Se, em um primeiro nível, dados são agregados de acordo com sua classificação conceitual definida por uma ontologia, em um nível abaixo podem ser relacionados de acordo com características existentes no seu conteúdo. Dessa forma o escopo da busca é reduzido pela busca conceitual e, em um próximo nível, uma busca por um dado específico e seus similares pode ser executada.

A metodologia apresentada nesta tese pode, inclusive, ser incorporada em sistemas que utilizam mais de um nível de abstração na classificação de seus dados. Esse tipo de abordagem é encontrada na literatura como, por exemplo, em [91] e [92]. Trabalhos desse tipo utilizam o conceito de ontologia superior (*upper ontology*) e ontologia inferior (*lower ontology*), sendo que a primeira define conceitos mais gerais e a segunda aborda aspectos mais detalhados dos elementos classificados, chegando mais próximo do conteúdo desses.

Como dito nas seções 6.2.2 e 6.4, outra característica que os cenários propostos nesta tese apresentam é a possibilidade do sistema identificar a existência de um mesmo conceito em mais de uma ontologia. Caso isso ocorra, uma busca por um conceito de uma ontologia pode ser completada com

[5] Não presente na Tabela 6.3.

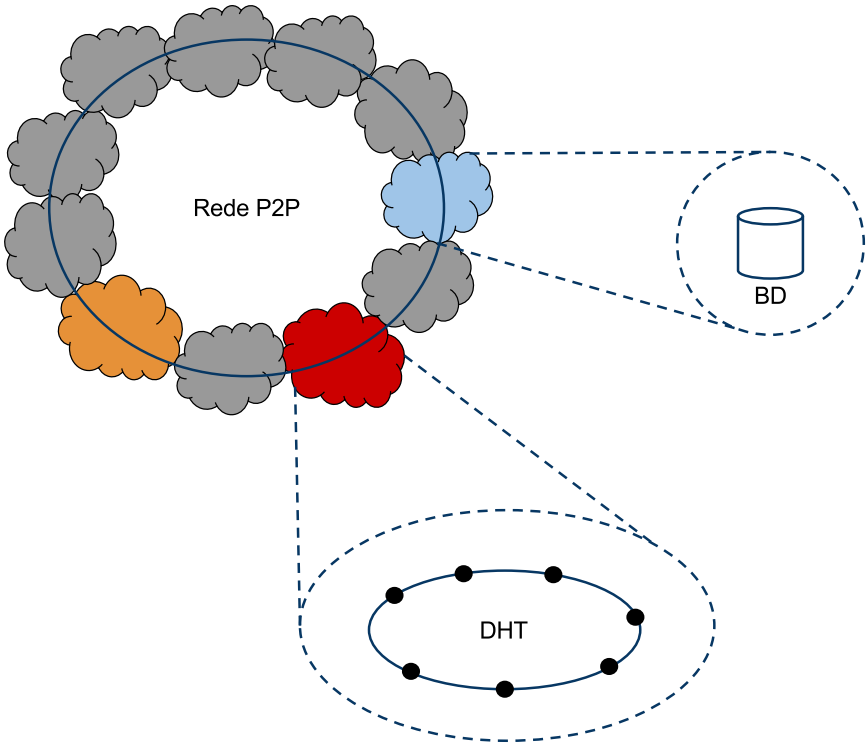


Fig. 6.29: Organização dentro das regiões.

dados classificados em outra ontologia que possua o mesmo conceito. No aspecto referente à busca por dados, esse fato possibilita novas formas de relacionar dados classificados por ontologias diferentes sem a necessidade da utilização de técnicas mais complexas, como a junção (*merge*) de ontologias, entre outras.

6.7 Sumário

Os capítulos anteriores apresentaram como é possível calcular a similaridade entre conceitos de uma mesma ontologia. Ao utilizar essa ontologia para classificar dados, é possível relacioná-los conceitualmente, calculando aqueles que são mais ou menos similares entre si. A partir da classificação conceitual dos dados é possível criar funções LSH que gerem valores de *hash* que mantenham a similaridade entre eles, do ponto de vista conceitual. Em um sistema de indexação, a utilização desse valor de *hash* para indexar os dados facilita a busca conceitual.

Este capítulo apresentou dois cenários de aplicação, no qual repositórios de dados classificados conceitualmente são indexados em uma rede P2P baseada em DHT. Foram feitos testes para comprovar a eficácia e os ganhos ao se utilizar funções LSH na indexação.

O primeiro cenário adapta a função LSH para agregar os dados na distância Euclidiana em um

espaço de endereçamento em forma de anel. Para isso foi criado um fator *m* que pode ser utilizado para aumentar ou diminuir a agregação da função LSH. Nesse cenário também foi explorada a ideia de se adicionar o vocabulário da ontologia no *hash*, na forma de um prefixo.

O segundo cenário mostra o uso da função LSH apresentada nesta tese, que mantém a similaridade dos dados pela distância de Hamming entre os identificadores gerados. Nesse segundo cenário, o espaço de indexação é dividido em regiões, cada uma representando um conceito de uma ontologia, interligados entre si seguindo o fenômeno do *Small World*.

Os testes mostraram que, em ambos os cenários, a função LSH agregou os dados e facilitou a busca por conceitos similares indexados em uma DHT.

Capítulo 7

Conclusão e trabalhos futuros

O ser humano utiliza diferentes tipos de conhecimento para lidar com informações. Uma das áreas de interesse na Ciência da Computação trata de como representar esses conhecimentos de forma eficiente, permitindo que máquinas possam processar e utilizar informações que, de certa forma, são triviais para os seres humanos. Para qualquer um desses conhecimentos e suas formas de representação em um ambiente computacional, o conceito de similaridade é muito importante. É por meio de comparações de similaridade com algo já conhecido que o ser humano consegue se adaptar a novas situações. Ao se definir um domínio de conhecimento, utilizando uma ontologia, por exemplo, o uso da similaridade entre conceitos dessa ontologia propicia o relacionamento entre eles, estabelecendo relações e novas informações.

Atualmente a Web Semântica vive um grande crescimento. Isso faz com que o volume de dados semanticamente anotados disponíveis na WWW cresça. É comum encontrar projetos que disponibilizam seu conteúdo de forma compatível com a Web Semântica, seguindo um domínio de conhecimento definido por uma ontologia simples, ou hierarquia de conceitos. Essa representação de um domínio que estabelece relações entre conceitos pode ser usada na indexação dos dados, facilitando a busca conceitual, na qual há o interesse de se obter dados e informações sobre um determinado conceito, e não simplesmente um dado específico.

Esta tese apresentou uma metodologia para representar o conhecimento humano em um valor binário por meio da criação de identificadores conceituais, criados por funções LSH, a partir de ontologias simples usadas para classificar dados. Dessa forma, diferentes tipos de dados, classificados de forma similar, podem ser relacionados entre si. Essa abordagem se mostrou proveitosa em cenários distribuídos de indexação, como nas redes P2P. Por meio desses identificadores, repositórios de dados similares foram indexados próximos uns aos outros na rede P2P, fazendo com que uma busca por conceitos similares fosse facilitada. A busca por um dado específico, levando-se em conta o conteúdo desse, também é facilitada, já que é possível limitar conceitualmente o seu escopo. Os

testes mostraram que, tanto em um cenário que leve em consideração a distância de Hamming entre os identificadores criados, como em um cenário que indexe dados no espaço Euclidiano, conceitos similares foram armazenados próximos uns dos outros.

Para trabalhos futuros, é possível citar alguns assuntos a serem pesquisados e aprofundados que podem dar continuidade às ideias aqui abordadas:

- investigar o relacionamento entre ontologias: no Capítulo 6 foi apresentada, na Seção 6.2, uma forma de se relacionar dados similares de diferentes ontologias, apresentando como criar um prefixo para as chaves resultantes a partir do vocabulário das ontologias. Um ponto possível de aprofundamento é a pesquisa sobre novas formas de se relacionar ontologias diferentes que definam o mesmo assunto, podendo levar em consideração análises mais profundas de comparação entre ontologias, encontradas na literatura, que consideram não só o vocabulário, mas também outros aspectos como o próprio domínio que essas definem;

- utilização de ontologias *heavyweight*: neste trabalho foram utilizadas ontologias simples, chamadas *lightweight*, as quais possuem somente relações simples e iguais entre os conceitos. Geralmente essas relações são do tipo “é um” e “é parte de”. Uma possível continuação do trabalho poderia investigar como a abordagem aqui apresentada pode ser utilizada para ontologias mais complexas, com diferentes tipos de relações entre os conceitos. Por exemplo, um conceito pode estar ligado a outro por meio de relações do tipo “emprega”, “é amigo de”, “pertence a”, “executa” e muitas outras. A utilização de diferentes valores de peso para as relações pode ser a primeira abordagem a ser testada;

- investigação de outros métodos de similaridade entre conceitos: utilização de métodos de similaridade entre conceitos de uma hierarquia que utilizem diferentes parâmetros na contabilização. Pesquisa sobre métodos que utilizem pesos nas relações, podendo diferenciá-las, e constatação de como isso pode impactar nas funções LSH. A partir dessa abordagem, é possível desenvolver técnicas para investigar a utilização de ontologias mais complexas;

- explorar novas formas de utilização de identificadores conceituais: esta tese apresentou como criar identificadores conceituais, utilizando funções LSH, mantendo a similaridade entre conceitos de uma ontologia. Os testes mostraram cenários de aplicação desses identificadores em redes P2P. Para trabalhos futuros, a exploração de novas maneiras de se utilizar esses identificadores pode ser aprofundada, como, por exemplo, na identificação de máquinas de acordo com seu conteúdo, em um esquema de identificação semântica para uma abordagem de roteamento plano [93], assim como em redes de conteúdo e *data centers* [94];

- explorar a possibilidade de utilização da abordagem aqui proposta em outras formas de definição de domínios, não somente estruturas hierárquicas, como as ontologias aqui mostradas. Por exemplo, conjunto de *tags* que não possuam uma ligação explícita entre si, não havendo uma relação hierárquica propriamente dita. Uma abordagem a ser pesquisada é a de conjunto de *tags* relacionadas;

- investigar estruturas de DHT baseadas em similaridade: na Seção 6.4 foi apresentada uma estrutura estática que utiliza o código de Gray e a distância de Hamming para organizar seus nós e *fingers*. Como mostrado, essa característica é vantajosa para a indexação de dados que possuam a distância de Hamming como índice de similaridade. Em trabalhos futuros, esse tipo de estrutura distribuída poderia ser melhor explorada, adicionando fatores dinâmicos e outras maneiras de organizar o espaço de indexação, aperfeiçoando ainda mais o seu rendimento em buscas por similaridade.

  Esta tese apresentou como utilizar a classificação conceitual dos dados na indexação destes. Dessa forma, dados com a mesma classificação conceitual, e mesmo outros com classificação similar, são armazenados próximos uns dos outros em um espaço distribuído de indexação, facilitando sua busca.

Referências Bibliográficas

[1] Artem Polyvyanyy. Evaluation of a novel information retrieval model: etvsm. Master's thesis, Hasso Plattner Institut, Univeristat Potsdm, 2007.

[2] Dragan Gasevic, Dragan Djuric, and Vladan Devedzic. *Model Driven Architecture and Ontology Development*. Springer, July 2006.

[3] A. Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977.

[4] *Mental Leaps: Analogy in Creative Thought*. The MIT Press, January 1996.

[5] R. L. Goldstone, D. L. Medin, and J. Halberstadt. Similarity in Context. , , 237-255. *Memory & Cognition*, 25:237–255, 1997.

[6] T. Mitchell. *Machine Learning (Mcgraw-Hill International Edit)*. McGraw-Hill Education (ISE Editions), 1st edition, October 1997.

[7] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan, New York, 1994.

[8] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:881–892, July 2002.

[9] Jane Hunter, John Drennan, and Suzanne Little. Realizing the hydrogen economy through semantic web technologies. *IEEE Intelligent Systems*, 19:40–47, January 2004.

[10] Valentina Cordì, Paolo Lombardi, Maurizio Martelli, and Viviana Mascardi. An ontology-based similarity between sets of concepts. In *Simulazione e Analisi Formale di Sistemi Complessi - WOA 2005*, pages 16–21, 2005.

[11] Luciano B. de Paula, Rodolfo S. Villaça, and Maurício F. Magalhães. Analysis of concept similarity methods applied to an lsh function. In *Computer Software and Applications Conference 2011 (Compsac 2011)*, Munich, Germany, July 2011.

[12] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. In *Scientific American*. May 2001.

[13] Lev Ratinov, Dan Roth, and Vivek Srikumar. Conceptual Search And Text Categorization. *Technical Report UIUCDCS-R-2008-2932, UIUC, CS Dept.*, 2008.

[14] Vic Lyte, Sophia Jones, Sophia Ananiadou, and Linda Kerr. UK Institutional Repository Search: Innovation and Discovery. *Ariadne Issue 61*, 2009.

[15] Anna Formica. Concept similarity in formal concept analysis: An information content approach. *Know.-Based Syst.*, 21(1):80–87, 2008.

[16] Vladan Devedzic. Knowledge modeling - state of the art. *Integrated Computer-Aided Engineering*, 8:257–281, 2001.

[17] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *Proceedings of ACM SIGCOMM'01*, pages 149–160, 2001.

[18] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60:327–336, 1998.

[19] Abhishek Gupta, Divyakant Agrawal, and Amr El Abbadi. Approximate range selection queries in peer-to-peer systems. In *CIDR*, 2002.

[20] Yingwu Zhu. *Enhancing Search Performance in Peer-to-Peer Networks*. PhD thesis, Computer Science and Engineering, University of Cincinnati, 2005.

[21] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, New York, NY, USA, 2002. ACM Press.

[22] Ondrej Chum, Michal Perdoch, and Jiri Matas. Geometric min-hashing: Finding a (thick) needle in a haystack, 2009.

[23] Brian Kulis, Prateek Jain, and Kristen Grauman. Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:2143–2157, 2009.

[24] Parisa Haghani, Sebastian Michel, Philippe Cudré-Mauroux, and Karl Aberer. Lsh at large - distributed knn search in high dimensions. In *WebDB*, 2008.

[25] Konstantinos Georgoulas and Yannis Kotidis. Random hyperplane projection using derived dimensions. In *Proceedings of the Ninth ACM International Workshop on Data Engineering for Wireless and Mobile Access*, MobiDE '10, pages 25–32, New York, NY, USA, 2010. ACM.

[26] M. Ryyndnen and A. Klapuri. Query by humming of midi and audio using locality sensitive hashing. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 2249–2252, 2008.

[27] K. Yoshida and N. Murabayashi. Tiny lsh for content-based copied video detection. In *Applications and the Internet, 2008. SAINT 2008. International Symposium on*, 28 2008.

[28] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 950–961. VLDB Endowment, 2007.

[29] Wang Weihong and Wang Song. A scalable content-based image retrieval scheme using locality-sensitive hashing. In *Proceedings of the 2009 International Conference on Computational Intelligence and Natural Computing - Volume 01*, CINC '09, pages 151–154, Washington, DC, USA, 2009. IEEE Computer Society.

[30] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *IEEE International Conference on Computer Vision (ICCV*, 2009.

[31] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 835–846, New York, NY, USA, 2006. ACM.

[32] Rudolf Wille. Restructuring lattice theory: An approach based on hierarchies of concepts. In *Proceedings of the 7th International Conference on Formal Concept Analysis*, ICFCA '09, pages 314–339, Berlin, Heidelberg, 2009. Springer-Verlag.

[33] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1997.

[34] S. Chu and B. Cesnik. Knowledge representation and retrieval using conceptual graphs and free text document self-organisation techniques. *Int J Med Inf*, 62(2-3):121–33, 2001.

[35] Judith P. Dick. Representation of legal text for conceptual retrieval. In *Proceedings of the 3rd international conference on Artificial intelligence and law*, ICAIL '91, pages 244–253, New York, NY, USA, 1991. ACM.

[36] Iadh Ounis and Marius Pasca. Modeling, indexing and retrieving images using conceptual graphs. In *In DEXA '98*, pages 24–28. Springer, 1998.

[37] Gi-Chul Yang and Jonathan Oh. Knowledge acquisition and retrieval based on conceptual graphs. In *Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing: states of the art and practice*, SAC '93, pages 476–481, New York, NY, USA, 1993. ACM.

[38] Gilad Mishne and Maarten De Rijke. Source code retrieval using conceptual similarity. In *Proc. 2004 Conf. Computer Assisted Information Retrieval (RIAO '04*, pages 539–554, 2004.

[39] B. S. Baker. On finding duplication and near-duplication in large software systems. In *Proceedings of the Second Working Conference on Reverse Engineering*, WCRE '95, pages 86–, Washington, DC, USA, 1995. IEEE Computer Society.

[40] Henrik Bulskov and Troels Andreasen. On measuring similarity for conceptual querying. In *In Proc. of the 5th International Conference on Flexible Query Answering Systems, Springer-Verlag publisher*, pages 100–111. Springer, 2002.

[41] Claudia D'Amato, Steffen Staab, and Nicola Fanizzi. On the influence of description logics ontologies on conceptual similarity. In *Proceedings of the 16th international conference on Knowledge Engineering: Practice and Patterns*, EKAW '08, pages 48–63, Berlin, Heidelberg, 2008. Springer-Verlag.

[42] Jacob Köhler, Stephan Philippi, Michael Specht, and Alexander Rüegg. Ontology based text indexing and querying for the semantic web. *Know.-Based Syst.*, 19:744–754, December 2006.

[43] Antonio M. Rinaldi. An ontology-driven approach for semantic information retrieval on the web. *ACM Trans. Internet Technol.*, 9:10:1–10:24, July 2009.

[44] Yingwu Zhu and Yiming Hu. Efficient semantic search on dht overlays. *J. Parallel Distrib. Comput.*, 67:604–616, May 2007.

[45] Habib Rostami, Jafar Habibi, and Emad Livani. Semantic routing of search queries in p2p networks. *J. Parallel Distrib. Comput.*, 68(12):1590–1602, 2008.

[46] Arturo Crespo and Hector Garcia-Molina. Semantic overlay networks for p2p systems. *Third International Workshop on Agents and Peer-to-Peer Computing - AP2PC*, pages 1–13, July 2004.

[47] Peter Haase, Ronny Siebes, and Frank Van Harmelen. F.: Peer selection in peer-to-peer networks with semantic topologies. In *In: International Conference on Semantics of a Networked World: Semantics for Grid Databases*, 2004.

[48] Yongcai Tao, Hai Jin, Song Wu, and Xuanhua Shi. Scalable dht- and ontology-based information service for large-scale grids. *Future Gener. Comput. Syst.*, 26:729–739, May 2010.

[49] Mario Schlosser, Michael Sintek, Stefan Decker, and Wolfgang Nejdl. HyperCuP - Hypercubes, Ontologies and Efficient Search on P2P Networks. *International Workshop on Agents and Peer-to-Peer Computing*, 2002.

[50] John Durkin. *Expert Systems: Design and Development*. Macmillan Coll Div, 1st edition, March 1994.

[51] John Mccarthy. Concepts of logical ai, 2000.

[52] John E Sowa and Stuart C. Shapiro. Knowledge representation: Logical, philosophical, and computational foundations computational foundations by john f. sowa (book review), 2006.

[53] H. Schrobe R. Davis and P. Szolovits. What is knowledge representation? *AI Magazine*, 14(1):17–33, 1993.

[54] W. J. Rapaport. Cognitive science. *Encyclopedia of Computer Science*, 2000.

[55] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5:199–220, June 1993.

[56] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21:96–101, May 2006.

[57] Michael Uschold and Michael Gruninger. Ontologies and semantics for seamless connectivity. *SIGMOD Rec.*, 33(4):58–64, 2004.

[58] Tim Berners-Lee and Mark Fischetti. *Weaving the Web : the past, present and future of the World Wide Web by its inventor*. Orion Business, November 1999.

[59] Eyal Oren. *Algorithms and Components for Application Development on the Semantic Web*. PhD thesis, National University of Ireland, Galway, 2008.

[60] G. Klyne and J. J. Carroll. Resource description framework: Concepts and abstract syntax. W3C Recomendation, 2004.

[61] Deborah L. McGuinness and Frank van Harmelen. Owl web ontology language. W3C Recomendation, 2004.

[62] Grigoris Antoniou and Frank vanHarmelen. *A Semantic Web Primer*. MIT Press, Cambridge, MA, USA, 2004.

[63] Mathieu d'Aquin, Enrico Motta, Marta Sabou, Sofia Angeletou, Laurian Gridinoc, Vanessa Lopez, and Davide Guidi. Toward a new generation of semantic web applications. *IEEE Intelligent Systems*, 23:20–28, May 2008.

[64] Wang Wei, Payam M. Barnaghi, and Andrzej Bargiela. Search with meanings:an overview of semantic search systems. *International journal of Communications of SIWN*, 3:76–82, 2008.

[65] Jacob Köhler, Stephan Philippi, Michael Specht, and Alexander Rüegg. Ontology based text indexing and querying for the semantic web. *Know.-Based Syst.*, 19(8):744–754, 2006.

[66] Douglas L. Medin, Robert L. Goldstone, and Dedre Gentner. Similarity involving attributes and relations: Judgments of similarity and difference are not inverse. *Psychological Science*, 1(1):64–69, 1990.

[67] Douglas L. Medin and Edward E. Smith. Concepts and Concept Formation. *Annual Review of Psychology*, 35(1):113–138, 1984.

[68] Gregg C. Oden. Concept, knowledge, and thought. *Annual Review of Psychology*, 38:203–227, 1987.

[69] Y. Li, Z.A. Bandar, and D. Mclean. An approach for measuring semantic similarity between words using multiple information sources. *Knowledge and Data Engineering, IEEE Transactions on*, 15(4):871–882, july-aug. 2003.

[70] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *Systems, Man and Cybernetics, IEEE Transactions on*, 19(1):17–30, jan/feb 1989.

[71] Philip Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, (11):95–130, 1999.

[72] Raquel Kolitski Stasiu. *Avaliação da Qualidade de Funções de Similaridade no Contexto de Consultas por Abrangência*. PhD thesis, Instituto de Informática, Universidade Federal do Rio Grande do Sul, 2007.

[73] Christoph Kiefer. *Non-deductive reasoning for the Semantic Web and Software Analysis*. PhD thesis, Faculty of Economics, Business Administration and Information Technology of the University of Zurich, 2008.

[74] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.

[75] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5:3–55, January 2001.

[76] Sheldon M. Ross. *A First course in probability*. Prentice Hall, 6. ed edition, 2002.

[77] P. Bouquet, G. M. Kuper, M. Scoz, and S. Zanobini. Asking and answering semantic queries. In *Workshop on Meaning Coordination and Negotiation Workshop (MCN-04) in conjunction with the 3rd International Semantic Web Conference (ISWC-04)*, Hiroshima, Japan, November 2004.

[78] C. Leacock and M. Chodorow. Combining local context and wordnet similarity for word sense identification. In *Fellbaum MIT Press*, 1998.

[79] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138, 1994.

[80] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, pages 154–165, 2009.

[81] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, New York, NY, USA, 1998. ACM.

[82] V. D. Chepoi. Isometric subgraphs of hamming graphs and d-convexity. *Cybernetics and Systems Analysis*, 24:6–11, 1988. 10.1007/BF01069520.

[83] J. P. Nolan. *Stable Distributions - Models for Heavy Tailed Data*. Birkhauser, Boston, 2011. In progress, Chapter 1 online at academic2.american.edu/∼jpnolan.

[84] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, SCG '04, pages 253–262, New York, NY, USA, 2004. ACM.

[85] V. M. Zolotarev. One-dimensional stable distributions. *Translations of Mathmatical Monographs, American Mathematical Society*, 65, 1986.

[86] Uldis Bojars, John G. Breslin, Vassilios Peristeras, Giovanni Tummarello, and Stefan Decker. Interlinking the social web with semantics. *IEEE Intelligent Systems*, 23(3):29–40, 2008.

[87] Rodolfo Villaça, Luciano de Paula, Mauricio Magalhães, and Fábio Luciano Verdi. Avaliação de redes p2p baseadas no fenômeno small world para o compartilhamento de conteúdos similares gerados por funções lsh. In *VI Workshop de Redes Dinâmicas e Sistemas P2P (WP2P 2010)*, Gramado, RS, May 2010.

[88] Sarunas Girdzijauskas. *Designing Peer-to-Peer Overlays: a Small-World Perspective*. PhD thesis, Ecole Polytechnique Federale de Lausanne (EPFL), Lausane, CH, 2009.

[89] Jeffrey Travers and Stanley Milgram. An Experimental Study of the Small World Problem. *Sociometry, Volume 32, Number 4*, pages 425–443, 1969.

[90] Sarunas Girdzijauskas, Anwitaman Datta, and Karl Aberer. Structured overlay for heterogeneous environments: Design and evaluation of oscar. *ACM Trans. Auton. Adapt. Syst.*, 5:2:1–2:25, February 2010.

[91] Tao Gu, Daqing Zhang, and Hung Keng Pung. A two-tier semantic overlay network for p2p search. *Parallel and Distributed Systems, 2007 International Conference on*, 2:1–8, Dec. 2007.

[92] Giuseppe Pirró, Domenico Talia, Paolo Trunfio, Paolo Missier, and Carole Goble. Ergot: Combining DHTs and SONs for Semantic-Based Service Discovery on the Grid. *CoreGRID Technical Report TR-0177*, August 2008.

[93] Rafael Pasquini. *Proposta de um mecanismo de roteamento plano baseado em operações de OU exclusivo e visibilidade local*. PhD thesis, Faculdade de Engenharia Elétrica e de Computação, FEEC, Unicamp, 2011.

[94] Christian Esteve Rothenberg. *Compact forwarding: A probabilistic approach to packet forwarding in content-oriented networks*. PhD thesis, Faculdade de Engenharia Elétrica e de Computação, FEEC, Unicamp, 2010.