J Braz Comput Soc (2011) 17:241–254
DOI 10.1007/s13173-011-0042-6

ORIGINAL PAPER

# Organization of multimedia data for conceptual search based on ontologies

**Luciano Bernardes de Paula · Rodolfo da Silva Villaça · Maurício Ferreira Magalhães**

Received: 14 February 2011 / Accepted: 19 September 2011 / Published online: 11 October 2011


**Abstract** Nowadays, there is a large volume of semantically annotated multimedia data available in the Semantic Web. These data have originated from several different sources, generating new issues about their storage and retrieval. In this scenario, simple ontologies are commonly used to define knowledge domains and classify data into concepts, establishing relations between them. Such conceptual relationship may be measured by a similarity function which allows the search to be performed by similarity in an indexing system. The contribution of this paper is to propose how to organize multimedia data using this conceptual classification in LSH (Locality Sensitive Hashing) functions, facilitating the conceptual search in distributed systems like P2P networks.

**Keywords** Semantics · Semantic web · Conceptual search · Small world

A previous version of this paper appeared at WEBMEDIA 2010, the Brazilian Symposium on Multimedia and the Web.

L.B. de Paula (✉) · M.F. Magalhães
DCA, FEEC, UNICAMP, Av. Albert Einstein, 400, Campinas, Brazil
e-mail: luciano@dca.fee.unicamp.br

M.F. Magalhães
e-mail: mauricio@dca.fee.unicamp.br

R.S. Villaça
DECOM/CEUNES, UFES, Rod. BR101N, km 60, São Mateus, Brazil
e-mail: rodolfovillaca@ceunes.ufes.br

## 1 Introduction

As the Semantic Web grows and becomes more popular, the volume of semantically annotated data available also increases [1]. Pictures, drawings, videos, movies, music are just a few examples of data that can be semantically classified [2]. This semantic classification is generally defined using standards like RDF[1] (Resource Description Framework) or OWL[2] (Web Ontology Language) and helps the machines to understand the meaning of data, not just by their type and content. For instance, a system may understand a picture just as a file that represents an image or as *"a picture of a beach"*, a video as *"a video of a football match"*, a text as *"a text about Brazilian politics"*, a website as *"a website of an enterprise"*, and so on.

Several works in the literature relate data just considering their contents. Images are related to each other by their visual characteristics (for example histogram and geometric shapes) [3, 4], texts are related by their keywords [5, 6], and others. Above the data content similarity there is another similarity level, in which data of several types can be grouped by their classification in a knowledge domain composed of a set of concepts. For example: a text about football, a video about a football match and a picture of a highlight moment of a football match can be grouped into the same concept, for instance, *Football*. Using this approach, a conceptual search [7–9] can be performed, in which the goal is to retrieve data classified into the same concept of a knowledge domain. This type of search is broader than searches based on keywords, as stated in [7]. In a keyword based search, the existence of those keywords or synonyms

[1] RDF standards: http://www.w3.org/RDF/ (accessed in July 3, 2011).

[2] OWL standards: http://www.w3.org/TR/owl-features/ (accessed in July 3, 2011).

in the data is necessary. In a concept search, regardless of the data contents, these can be classified into a specific concept of the domain. A query about a concept of a knowledge domain, for example the concept *Football*, can be answered with any data classified into that concept. Even a search for specific data may be facilitated as the scope of the search is reduced just to data that are conceptually similar.

Conceptual search relies on a definition of a knowledge domain. One of the ways to organize concepts of a domain is using simple ontologies (also known in the literature as lightweight ontologies, *IS-A* hierarchy of concepts, and others). These simple ontologies define the relationship between the concepts, relating them to each other. Following the previous example, the concept *Football* may be a subconcept of the concept *Sports*, both belonging to an ontology that defines the domain *Entertainment*.

In the Semantic Web scenario, several projects use the semantic classification of a hierarchy of concepts for several different domains. For example, *MusicMoz*,[3] *Musicbrainz*,[4] and *Yahoo!*[5] are some examples that have a definition for the domain *Music* using concepts.

Several works in the literature analyze the similarity between concepts of an ontology, as in [10] and [11], presenting methods to determine a value to relate them. Similarity between concepts can be used in a conceptual search to expand the results. Depending on the interest of the user, a search about a concept may be completed with data classified into similar concepts. But none of these works consider this data similarity at the moment that they are stored and retrieved. If the storage of data is done considering their semantic classification the retrieval of semantically similar data would be facilitated.

On the other hand, some works deal with the storage and retrieval of data taking into account just their contents rather than their classification [3, 5]. For instance, images are considered similar by their contents and not by their classification in a knowledge domain. These works present improvements to similar data searching with no concern about the conceptual relation between them.

The contribution of this work is to explain how to create identifiers for sets of multimedia data using locality sensitive hash functions and keeping their conceptual similarity. This allows the relation of data with others that are similar on the conceptual level, facilitating a search in a distributed environment, such as a P2P network. After that, the use of these identifiers in the indexing of data in a structured P2P network is proposed. Such identifiers are distributed over two different Distributed Hash Tables (DHTs), Chord DHT and a Small World based P2P network. Also the effort needed to recover similar data in either structure is compared. Therefore, the intention is to facilitate the search for data classified into similar concepts.

The rest of this paper is organized as follows: Sect. 2 presents some related work and discusses the contributions of this work related to others from the literature. Section 3 defines some basic concepts used in this work and presents the application scenario. Section 4 explains how to extract the similarity level between concepts belonging to an ontology. Section 5 introduces the Locality Sensitive Hash (LSH) functions, presenting two examples, and in Sect. 6 a LSH function is proposed that uses the similarity between concepts in an ontology. Section 7 presents the Small World phenomenon and explains how it is possible to take advantage of it to create a DHT to facilitate the retrieval of similar data. Section 8 shows the results obtained in some tests to validate the proposal; Sect. 8.1 discusses how the proposal of this paper may be used in a real application using some of the Semantic Web standards, and Sect. 9 concludes the paper.

## 2 Related works

The related works may be divided into three categories:

- works that use LSH functions to relate data;
- works that use lightweight ontologies to define knowledge domains and perform conceptual search;
- works that use conceptual classification in P2P networks.

The next subsections discuss each of these categories and present some works representing them.

### 2.1 Use of LSH functions

LSH functions are used in several works to relate and compare similar data with such a similarity defined by a similarity function.

The authors of [3, 4] and [21] use LSH functions to relate and index images. The images are compared using characteristics such as color tones, textures, and visual words. The use of these image characteristics in a LSH function results in the same, or at least close indexing keys for similar images. Similar images are stored close to each other in an indexing system, facilitating the search for images with similar contents.

LSH functions are also used for other types of data: data from sensor networks [22], audio [23] and video files [24] are just a few examples found in the literature. In all these

[3] MusicMoz: concepts hierarchy (http://musicmoz.org, accessed in July 1, 2011).

[4] Musicbrainz: related tags (http://musicbrainz.org, accessed in July 1, 2011).

[5] Yahoo: directory of categories (http://dir.yahoo.com, accessed in July 1, 2011).

works, LSH functions are used to relate data to each other to make the search for similar data easier.

A characteristic of the works presented in this subsection is the use of LSH functions to relate data by their contents. This means that in a search for similar images, just the ones sharing similar visual characteristics are considered similar. As another example, in a search for texts, just the ones sharing the same keywords or vocabulary are going to be considered similar. However, a basic aspect of all these works is that none of them permits us to relate data of different types, for example an image and a text, even if both are related to a particular subject.

This paper proposes the use of LSH functions to represent concepts of an ontology. Using this approach, data may be related to each other by their conceptual classification in a knowledge domain.

### 2.2 Use of lightweight ontologies and conceptual search

The conceptual search is explored in several works in the literature. The goal of this search method is to retrieve information about a concept and not specific data. A conceptual search is considered more advantageous to find sets of information about some concepts if compared to the traditional keyword-based searching techniques, as indicated in [7]. Conceptual search may even be combined with traditional keyword searching [25].

In the literature, several works use a conceptual search to retrieve data in different types of scenario, like medical data [26], texts referring to laws and legal cases [27], images contents [28], source codes [29] and Semantic Web data [30]. Conceptual graphs are also used when the information sought does not exist [31].

In [8] are described the advances made by the *Institutional Repository Search Project*[6] about information retrieval in large repositories. The paper presents the experience and success obtained by the project using a conceptual search.

A characteristic of the works referenced in this subsection is that none of them discusses how to take advantage of the conceptual relation of data (texts, photos, videos, podcasts, etc.) while storing them in order to facilitate their retrieval. For example, data (or the respective links) that are conceptually similar may be indexed close to each other in a distributed system, facilitating their retrieval. The main objective is to minimize the number of hops needed to obtain conceptually similar data repositories decreasing both network traffic and the overall response time. The conceptual classification of data is used to aggregate them in the indexing space.

[6]Found at http://www.intute.ac.uk/irs (accessed in June 27, 2011).

### 2.3 Conceptual classification in P2P networks

Peer-to-Peer (P2P) overlay networks are recognized to facilitate the sharing of large amounts of data in a decentralized and self-organizing way. These networks offer valuable benefits for distributed applications in terms of efficiency, scalability, and resilience to node failures. Distributed Hash Tables (DHTs), for example, allow for efficient key lookups in a logarithmic number of routing hops but are typically limited to exact or range queries [3]. The following works have employed some form of semantic knowledge of data while indexing in distributed systems, as P2P networks.

In [17] and [32], the authors apply LSH functions to index documents in a P2P network. The former proposes a P2P data sharing architecture for computing approximate answers for the complex queries by finding data ranges that are similar to the user query instead of exact lookup operations; the latter proposes an approach to semantic search on DHT overlays. The basic idea is to place indices of semantically close files into the same peer nodes with high probability by exploiting information retrieval algorithms and locality sensitive hashing. Thus, similar data have great probability to have similar identifiers. In order to raise this probability each text in the paper is indexed at about 20 times in the P2P network.

Differently from DHT P2P networks which try to efficiently route messages from one peer to another, the Semantic Overlay Networks (SONs) try to find the right peers to answer a query. The node contents are considered to form SONs and interlink them. The idea is to appropriately route the queries increasing the chances that matching files will be found quickly [33].

The authors of [34] propose a model in which peers advertise their expertise in the P2P network. Knowledge of the expertise of other peers forms a semantic topology. Based on the semantic similarity between the subject of a query and the expertise of other peers, a peer can select appropriate peers to forward queries improving the performance of a P2P. Another approach introduces a topology based on a hypercube graph to cluster peers in a P2P network [35]. A global ontology is used to categorize the data from peers and to route queries. For each concept in the ontology a hypercube is built containing the peers supporting this concept.

In the papers discussed in this section a topology based on an ontology [35] is built or the peers have to advertise their expertise [33, 34]. The increase of the number of peers augments the cost and complexity for both approaches. The strategy adopted in our proposal is the simplification of the way the semantic concepts are utilized for the indexing and retrieval of documents. Instead of linking the peers with semantic fingers, we opted by clustering semantically similar documents in the indexing space represented by a DHT P2P network.

Also a characteristic of the works presented in this section is that, although they index data in a P2P environment and some of them use the conceptual classification of data to route messages, none of them uses the conceptual classification of data to organize their storage.

This work explains how to use the conceptual classification to aggregate similar data in an indexing space, without the need of key overloading but still facilitating the retrieval of similar and related data.

The next section presents the basic concepts and scenarios that organize data classified by ontologies in a distributed system, facilitating the conceptual search.

## 3 Basic concepts and scenario

The next subsections present the basic concepts of this paper and an application scenario.

### 3.1 Basic concepts

Before presenting the proposal for data organization by their conceptual classification, some elements that compose the scenarios of this work are defined:

- *data*: correspond to any structure that can be stored and retrieved in a computational system. Images, texts, audio files, webpages among others are examples of data;
- *metadata*: annotations appended to data defining their classification based on a concept hierarchy, taxonomy, ontology, and so on. Generally these annotations are written in RDF. In this text the term *data* is used for the data themselves as well as for their metadata;
- *data repositories*: every machine (or set of machines) that shares data or metadata of several types, semantically annotated following some classification. Possible examples: personal computers with music and video files, a group of data servers of a university with multimedia data, RDF file repositories of Web Semantic, etc..
- *ontology*: in the context of this paper, ontology consists of a data structure that defines a domain, composed of concepts and their relations, allowing the reasoning and the inference about some concept of the ontology and the relation with other concepts [12]. In this paper, simple ontologies are used, which are also known in the literature as lightweight ontologies and concept hierarchies *IS-A*, among others, in which the relations between concepts are of the types *"is a"* and *"part of"*. In the rest of this text, the word *ontology* refers to this structure.

These elements are used in the application scenario presented in the next subsection.

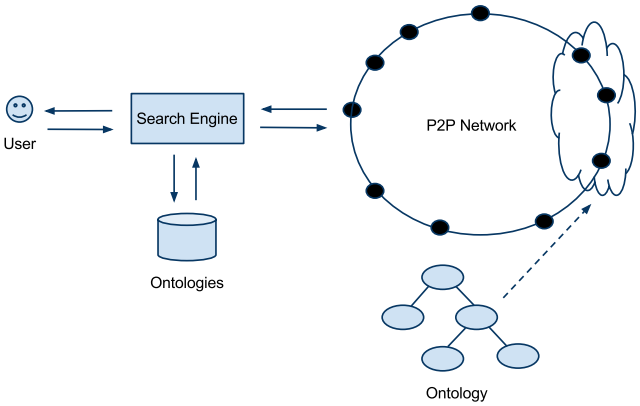


**Fig. 1** System organization—the user submits a query about a concept of an ontology to a search engine; a key representing this concept is looked for in the P2P network, returning the data indexed; similar concepts are indexed close to each other in the P2P indexing space, facilitating a search by similarity

### 3.2 Application scenario

Figure 1 illustrates the proposal of data organization presented in this paper for a conceptual search. The proposal is described below.

Several repositories of an enterprise or a university provide multimedia data (images, texts, audio, videos, and others) about several subjects. Each repository uses simple ontologies that define knowledge domains, with dozens or hundreds concepts, to classify its data. The ontologies are defined using OWL and are stored in a repository.

These repositories hold conceptually classified data and are indexed in a structured P2P network, using DHT (Distributed Hash Table). Indexing is done considering the similarity between concepts of the ontologies. In order to accomplish this, each concept is represented by a concept key created using LSH functions. This approach results in similar data being indexed near each other in the DHT indexing space, facilitating the retrieval of groups of similar data by the reduction of the effort in terms of the number of hops needed to perform such action.

As proposed in this paper, a scenario of a conceptual search can be illustrated by a user consulting a system, indicating the concept that he or she wants to research about. The determination of this concept can be done:

(i) in an explicit way, in which the user indicates which concept of an ontology he or she is interested in to search; or
(ii) in an implicit way, in which the concept searched and the ontology are determined by natural language analyzers and reasoners.

These ontologies are obtained in an ontology repository. The implicit form is beyond the scope of this work. Still at the moment of the search, the user may also choose a level of similarity, represented by a value between 0 and 1, that

he or she allows to be used in the answer provided (or this value may be intrinsic to the system).

The searched concept is translated to a concept key that was previously indexed in the P2P network. This concept key is searched in the P2P networking, returning to the user the information stored with it. This information may be the locator of the repositories that have data classified into that concept or a metadata pointing to a URI from where it is possible to retrieve them. Data that were inserted using keys of similar concepts may also be returned, according to the similarity level used in the search. The result of the search may be used by another service, responsible for filtering all obtained data, to best fit the needs of the user; for example, if the user wants to restrict the results by type like just images, just texts, etc..

A common procedure to create the keys to be used in P2P networks is to use a hash function, for example MD5 (Message-Digest algorithm 5) or SHA-1 (from the family SHA of hash functions). The value returned by a hash function is the identifier of that content in the P2P network, used to store and retrieve data. Generally, structured P2P networks are built on top of DHTs in which data are inserted by the primitive $put(k, v)$ and retrieved using $get(k)$, in which $k$ is the key used as the identifier of the data and $v$ is the value related to that identifier, e.g. data themselves, metadata, a locator of where data are actually located, etc.. There are several ways to organize nodes in a DHT like a ring, a mesh or a tree, for example. In this work DHTs based on Chord [13] and on a Small World topology [20] were considered. Nodes are organized into a virtual ring on both cases, but they have different routing strategies.

A characteristic of the usage of ordinary hash function in the creation of the keys to identify data is that there is no semantic relation between the value returned by the function and the data that originated it. Two similar data generate two unrelated hash values while applied to such hash functions. This leads to situations in which it is costly to retrieve similar data, from the P2P network point of view, as they are all spread in the indexing space. An option is to use Locality Sensitive Hash (LSH) functions [14, 15]. These functions, always bound to a similarity function, generate values that carry the similarities between the data that originated them. Thus, similar data have a great probability of generating similar or even the same hash values in some metric.

As discussed in Sect. 2, some works in the literature, like [5] and [3], are concerned with maintaining the relation between semantically similar data when stored into some indexing system, be it a centralized (database) or a distributed system. Even in these works the conceptual search is difficult as the LSH functions are bound to the similarity of the content of data to generate the keys and not to their conceptual classification.

As the focus of this paper is the conceptual search, the conceptual classification is used to generate an LSH function bound to the concept similarity. This way, similar concepts are stored close to each other in the virtual space in a P2P network, facilitating the search for related data. Figure 1 shows that the concept keys of an ontology are stored into the same region of the virtual space.

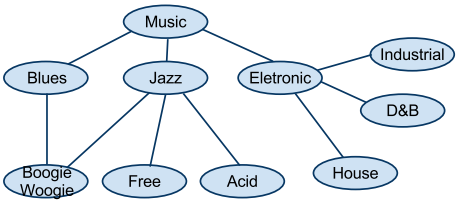


**Fig. 2** Lightweight ontology for the domain *Music*

The next section presents how to extract similarity between concepts of a simple ontology. This similarity is the base of the proposed LSH function.

## 4 Similarity based on ontologies

Several works in the literature, like [10] and [11], present methods to extract the similarity from ontologies. For instance, [16] uses a hierarchy based on WordNet,[7] an ontology of nouns of the English language, to describe texts.

This paper deals with the conceptual search, in which the data are classified according to what they represent, and not to their contents. Figure 2 presents an ontology based on some concepts of the directory of categories from *Yahoo!* for the *Music* domain. This classification may be used to index texts, videos, images about music, or even audio files. In this work, this concept hierarchy is used to define the similarity between concepts and this similarity is used to create an LSH function to relate concepts and facilitate their indexing and retrieval.

There are several methods in the literature for calculating the similarity between concepts of an ontology, as the ones presented in [10]. Each method takes into account different characteristics of the ontology, changing how the similarity of the concepts is defined. This paper uses the algorithm presented in [12] to generate concept vectors for each of the concepts of an ontology. This method is a natural choice to use with a LSH function, discussed in Sect. 5, because it generates a vector for each concept, while others just return a similarity value for each pair of concepts. The extraction of the concept vectors is explained below.

For all leaf concepts of a lightweight ontology, as the one presented in Fig. 2, a concept vector $\overrightarrow{\tau_i} = \{\tau_{i,1}, \tau_{i,2}, \ldots, \tau_{i,k}\}$ is created, in which $k$ is the total number of the ontology

[7] Wordnet: http://wordnet.princeton.edu (accessed on July 1st, 2011).

concepts. The concept vector is defined as follows:

$$\tau_{i,k} = \begin{cases} 1 & \text{if } \tau_k \in \text{ in the path from } \tau_i \text{ to root or } i = k \\ 0 & \text{otherwise.} \end{cases}$$

For all internal concepts of the ontology, the concept vector is calculated in the following manner:

$$\overrightarrow{\tau_i} = \left| \sum \overrightarrow{\tau_s} \right|$$

in which $\overrightarrow{\tau_s}$ are all concept vectors of the direct children of $\tau_i$. For all concepts of the ontology of Fig. 2, the normalized and approximated vectors are:

$\overrightarrow{Music} = (0.723, 0.279, 0.459, 0.263, 0.279, 0.09, 0.09, 0.087, 0.087, 0.087)$;

$\overrightarrow{Blues} = (0.5, 0.5, 0.5, 0, 0.5, 0, 0, 0, 0, 0)$;

$\overrightarrow{Jazz} = (0.642, 0.194, 0.642, 0, 0.194, 0.224, 0.224, 0, 0, 0)$;

$\overrightarrow{Eletronic} = (0.654, 0, 0, 0.654, 0, 0, 0, 0.218, 0.218, 0.218)$;

$\overrightarrow{Industrial} = (0.577, 0, 0, 0.577, 0, 0, 0, 0, 0, 0.577)$;

$\overrightarrow{BW} = (0.5, 0.5, 0.5, 0, 0.5, 0, 0, 0, 0, 0)$;

$\overrightarrow{Free} = (0.577, 0, 0.577, 0, 0, 0.577, 0, 0, 0, 0)$;

$\overrightarrow{Acid} = (0.577, 0, 0.577, 0, 0, 0, 0.577, 0, 0, 0)$;

$\overrightarrow{House} = (0.577, 0, 0, 0.577, 0, 0, 0, 0.577, 0, 0)$;

$\overrightarrow{DB} = (0.577, 0, 0, 0.577, 0, 0, 0, 0, 0.577, 0)$.

One common manner found in the literature to measure the similarity of vectors, and also used in [12], is by measuring the cosine of the angle between each pair of vectors. The larger the cosine value is, the more related the concepts are. In other words, if the angle between the vectors is small, they are more similar and vice versa. Table 1 presents the similarities calculated by the cosine of the angle between the vectors representing the concepts *Jazz* and *Free* with all others.

Using the previous procedure for lightweight ontologies, identification of which concepts are more or less similar to each other is possible. The definition of the similarity level between concepts allows that, for example, in searches of data classified as *Jazz*, if it is necessary to have an exact answer, this may be composed just of elements classified into this concept. However, if there is a tolerance, for instance, of 0.8 or more of similarity, the search may be answered with data classified as *Acid*, *Free*, *Blues*, *Boogie Woogie* and even *Music*, if broader concepts are allowed in the answer. This similarity can also be used to rank the results.

The similarity between the concepts defined by the method proposed in [12] privileges generalization instead of specialization. For example, the concept *Free* is more similar to concepts that are more generic (*Jazz* and *Music*) than the ones that are more specialized; this is so even for its siblings (*Boogie Woogie* and *Acid*).

**Table 1** Cosine between some of the concept vectors

| | $\overrightarrow{Jazz}$ | $\overrightarrow{Free}$ |
|---|---|---|
| $\overrightarrow{Music}$ | 0.908 | 0.735 |
| $\overrightarrow{Blues}$ | 0.836 | 0.577 |
| $\overrightarrow{Jazz}$ | 1 | 0.870 |
| $\overrightarrow{Elet}$ | 0.420 | 0.377 |
| $\overrightarrow{BW}$ | 0.836 | 0.577 |
| $\overrightarrow{Free}$ | 0.870 | 1 |
| $\overrightarrow{Acid}$ | 0.870 | 0.666 |
| $\overrightarrow{House}$ | 0.370 | 0.333 |
| $\overrightarrow{DB}$ | 0.370 | 0.333 |
| $\overrightarrow{Indust}$ | 0.370 | 0.333 |

It is important to notice that, by the form that the extraction of concept vectors of the lightweight ontologies is done, for the ontology of Fig. 2, the concept *Blues* and its subconcept *Boogie Woogie* have similarity equal to 1. This happens due to the fact that the more generic concept *Blues* has just one subconcept, which is responsible to relate them (considering lightweight ontologies with simple relations like *"is a"*).

The next section presents the LSH functions. These functions keep, with a certain probability, data similarity in the values of the identifiers generated, which will be used to store and retrieve the data in a DHT.

## 5 Locality Sensitive Hash functions (LSH)

This section deals with how LSH functions are defined and how they are used in the literature.

In [14], definitions are given such that a family of hash functions $F$ is classified as locality sensitive (LSH), corresponding to a similarity function $\text{sim}(a, b)$, if for all function $h \in F$, we have

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

in which **Pr** is the probability and $\text{sim}(a, b)$ is a similarity function that returns a value between 0 and 1; 1 for $a$ and $b$ completely related, and 0 otherwise.

Generally, LSH functions are used to index data, like images, texts and others, keeping their similarity, such a similarity being measured by a similarity function. LSH functions also represent data reducing their dimensions without losing their similarities, as explained in [14].

The next subsections present some examples of similarity functions and their corresponding LSH functions.

### 5.1 Min-wise independent permutations

The Min-wise independent permutations [5, 17] generates a family of LSH functions corresponding to the Jaccard similarity, defined by

$$\text{sim}_{\text{Jaccard}}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

in which $A$ and $B$ are set of integers.

Being $\pi$ a random permutation in the integer universe $I$, $A = \{a_1, a_2, \ldots, a_n\} \subseteq I$, and $B = \{b_1, b_2, \ldots, b_n\} \subseteq I$, the hash function $h_\pi$ is defined by

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

in which $h_\pi(A)$ applies the permutation $\pi$ in each element of $A$ and considers only the smaller one as the result. For two sets $A$ and $B$, we have $x = h_\pi(A) = h_\pi(B)$ only if $\pi^{-1}(x) \in (A \cap B)$. The probability defined for two sets $A$ and $B$ is

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

For example, in [32] the Min-wise function is used to index texts. Each text is represented by a set of keywords, which results in similar texts having similar sets of keywords. As these sets are used in a Min-wise function, similar texts have high probability of generating the same hash value.

### 5.2 Random Hyperplane Hash functions

Another similarity function that can be used is the cosine of the angle between two vectors, as presented in Sect. 4. $\theta$ is the angle, in radians, between the vectors $\overrightarrow{u}$ and $\overrightarrow{v}$. If $\cos\theta(\overrightarrow{u}, \overrightarrow{v}) = 1$, both vectors are completely related, and if $\cos\theta(\overrightarrow{u}, \overrightarrow{v}) = 0$, there is no relation between them.

A family of LSH functions that corresponds to the cosine is the Random Hyperplane Hash (RHH) function [15]. Given a collection of vectors in $R^d$, a random vector $\overrightarrow{r}$ is chosen from a d-dimensional Gaussian distribution (each coordinate is drawn from a Gaussian distribution, one 1-dimensional—a normal distribution). For this vector $\overrightarrow{r}$, the hash function is defined by

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

For vectors $\overrightarrow{u}$ and $\overrightarrow{v}$, according to [15] we have

$$\mathbf{Pr}[h_r(\overrightarrow{u}) = h_r(\overrightarrow{v})] = 1 - \frac{\theta(\overrightarrow{u}, \overrightarrow{v})}{\pi} \quad (2)$$

showing the relation between (2) and $\cos\theta(\overrightarrow{u}, \overrightarrow{v})$. As seen, the RHH function produces a single bit. In order to create longer keys, it is necessary to produce several vectors $\overrightarrow{r}$ and append the results, composing a binary string. For each position in this binary string, similar vectors have a high probability of producing the same result.

The authors of [4] use the RHH function to index images keeping the similarity between them, classifying and clustering them. The characteristics of the images are used to represent them as vectors like color tones and geometric shapes and the vectors are used in RHH. Similar images have similar vectors generating similar hash values.

The next section presents how to generate an LSH function based on the similarity between concepts of an ontology, combining the functions described above.

## 6 LSH function based on ontology

The creation of the keys is done in two steps. First, for a lightweight ontology, the concept vectors are extracted using the technique presented in Sect. 4. After that, each concept vector is used in a LSH function, generating a hash value for each concept of the ontology. The LSH function generates concept keys keeping the similarity between the concepts. The next subsection presents more details.

### 6.1 Creation of the keys

The generation of the concept keys is done by extracting the concept vector for all concepts in an ontology and applying the RHH function (Sect. 5.2) using these vectors. For the RHH function, $n$ vectors $\overrightarrow{r}$ are generated, each coordinate of these values is drawn from a normal distribution, with average 0 and variation 1. A key of $n$ bits is obtained for all concepts by calculating the scalar product of vectors $\overrightarrow{r}$ and concept vectors. All $n$ results are appended composing the key.

As dealt with in Sect. 5.2, two concept keys have the probability $p$ of having the same value for bits in the same position, $p$ being determined by the cosine of the angle between the concept vectors. This fact implies that, even for two keys of two very similar concepts, there is a probability that they will be stored far from each other in the linear virtual space of a DHT. This happens if a pair of bits in the same position of the most significant part of the keys does not have the same value. Trying to reduce this characteristic, our strategy consists of generating several keys to each concept and using the one with the smaller lexicographic value. In other words, the key that has the larger prefix with "0"s is chosen as the concept key. The probability is high that the smaller keys created are the same or have near values for similar concepts. This can be explained by the Jaccard similarity used in Min-wise: each group of $m$ concept keys

can be seen as a set of integers, and the probability of similar concepts to have the same small value is the same, as explained in Sect. 5.1.

Several works in the literature that use LSH functions, as [5] and [17], index data several times in the DHT. This is done in the attempt to augment the probability of two identifiers of similar concepts to be stored near each other but resulting in the necessity of having several lookups to retrieve information. Considering the increase of the volume of semantically annotated data available on the Web, this approach is not the best one, due to the overload of keys in the system and the number of lookups done in the system. In the proposal presented in this paper, although several keys are created for each concept, just the smaller one is used to index data. Using this procedure, there is no overload in the system if the volume of data indexed grows and, for each new concept indexed, just one key is inserted into the system. Our proposal of LSH function can be defined as follows:

- for each concept of a lightweight ontology, a concept vector is extracted;
- $m$ groups of $n$ vectors $\overrightarrow{r}$ are created, each coordinate is drawn from a normal distribution;
- for each concept vector, the function (1) is applied to each vector $\overrightarrow{r}$ of a group, generating a key of $n$ bits. This process is done $m$ times, once for each group of $m$ vectors.
- The smaller key among all $m$ keys created is chosen as the concept key.

It is important to explain the relation between $m$ and the similarity in the creation of the keys. If $m$ is equal to 1, the similarity of each pair of concept vectors, measured by the cosine of the angle between them, is preserved by the Hamming similarity between the created keys. For two strings of the same size $s_a$ and $s_b$, the Hamming similarity is measured as follows:

$$\text{ham}_{\text{sim}}(s_a, s_b) = \frac{\text{number of bits matching}}{\text{length of the keys}}.$$

Another equivalent measure is the Hamming distance, which is measured by counting the number of bits that does not match in two strings. The higher the Hamming similarity between two strings is, the closer they are in the Hamming distance.

The similarity is kept by the RHH function, which maintains the cosine similarity in the resulting hash value. As the RHH returns just one bit, to create a key with length $n$ it is necessary to append several results of RHH functions. Similar vectors, i.e. vectors with high cosine similarity, have a great probability to have the same value for each bit. Therefore, in the resulting key, the higher the similarity between the concept vectors is, the higher is the Hamming similarity between the keys, and the closer they are to each other in terms of the Hamming distance. In an indexing space based on the Hamming distance, for instance a hypercube, keys of similar concepts are stored near each other, so this is an indication to use $m$ equal to 1. Also, for any pair of keys, their Hamming distance shows their similarity.

On the other hand, if the indexing space is Euclidean, as commonly used in DHTs, using keys that are similar in the Hamming distance is not the best choice. For example: if two keys of 128 bits length match 127 bits and the bit that does not match is in the most significant part of the key, in a linear indexing space they are stored far from each other.

In order to address this problem $m > 1$ is used. As defined before, $m$ groups of vectors $\overrightarrow{r}$ are created, and the smaller key generated is chosen as the concept key. Doing this procedure, there is a great probability that similar concepts generate the smaller key from the same group of vectors $\overrightarrow{r}$ among the $m$ groups. But concepts that are not similar generate the keys from different groups of vectors $\overrightarrow{r}$, which means that, for any pair of keys, their Hamming similarity is lost, as they have might been created by different groups of vectors $\overrightarrow{r}$.

However, since the purpose of this paper is to aggregate the keys in an Euclidean indexing space, it is not necessary to keep the Hamming similarity, but store the keys of the same ontology near each other. For this purpose, the usage of higher values of $m$ is accepted. For these values of $m$, the concepts are stored closer to each other. Due to the necessities of the indexing system, $m$ may be adjusted to have more or less aggregation. In Sect. 8 the results are presented.

A problem of creating keys using this approach is that it considers just the topology of the ontology, i.e. two ontologies that have the same topology, but define two totally different domains, generate the same concept keys. The next subsection shows how to add the vocabulary of the ontology in the creation of the identifiers, avoiding that just the topology of the ontology is used to determine the index value of the concept. Basically, the keys are composed of two parts: a prefix that considers the vocabulary of the ontologies, and a suffix that considers the topology of the ontology.

### 6.2 The creation of prefixes

The usage of prefixes in the keys aggregates ontologies that have similar vocabulary and index ontologies that have the same topology but define different domains far from each other. At the same time, it brings about two ontologies that have different topologies but define the same domain close to each other. In order to accomplish this, the proposal is to compose concept keys in two parts. The first is a prefix generated from the vocabulary of the ontology, created by an LSH function. The suffix is obtained as presented in Sect. 6.1.

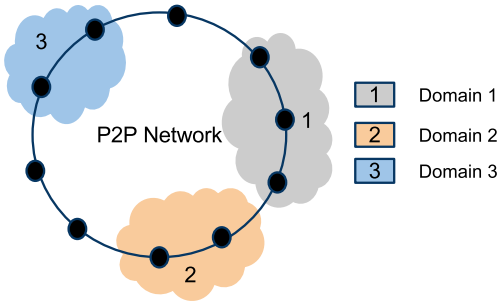


**Fig. 3** Domain distribution in a P2P network

The prefix is created using the Min-wise LSH function (Sect. 5.1). This function is applied to a vocabulary vector, which is created for each ontology and it is composed of the names of all concepts. Two similar ontologies have great chances of having the same prefix, given a similarity between their vocabularies. The generation of the prefix is done as follows:

- For all ontologies, a vocabulary vector $v$ is created, composed of the names of all concepts presented on the ontology;
- $v$ is converted in an integer vector $v'$ using a traditional hash function (like MD-5 or SHA-1) for each element of $v$;
- the vector $v'$ is used in a Min-wise function, as described in Sect. 5.1;
- the result of the Min-wise of $v'$, in other words, the smaller integer generated after the permutations, is used as the prefix in all keys of all concepts of that ontology.

Using this procedure, similar ontologies that have a similar vocabulary, i.e. define the same domain, for instance *Music*, generate the same prefix and are stored into the same region of the virtual space of the DHT. Ontologies that define other domains, for instance *Sport*, are stored into other regions of the virtual space, balancing the indexing space of the DHT. Figure 3 shows a possible distribution of three different domains in a P2P network.

This indexing space configuration can even enable a query about a concept of an ontology being answered with data classified into another ontology that defines the same knowledge domain. As data classified into ontologies of the same knowledge domain are stored into the same region of the indexing space, such a feature is facilitated in the system.

The next section presents the Small World phenomenon and how it can be applied in P2P networks, facilitating the search and recovery of semantically aggregated data.

## 7 The Small World phenomenon

As part of the studies developed in this research, new manners to organize a DHT were considered. This section discusses the Small World phenomenon, showing how to use it to organize the nodes of a DHT in order to facilitate the search for similar data.

The earlier studies involving the characteristics of this phenomenon, popularly known as the "six degrees of separation", were developed in the 1960s. Since then, many others explained and modeled this phenomenon by means of graphs and algorithms [18, 19]. We can say that a population has the characteristics of the Small World phenomenon if, for any two individuals from this population, there is a short path which links both, through a sequence of other individuals who have some common knowledge.

A model for building networks based on the Small World phenomenon was proposed by Kleinberg [19]. It divides the neighbors of a node into two types: local and long distance neighbors, and the long distance neighbors establishment should be achieved inversely proportional to some distance metric. Every node has the majority of their fingers pointing to others near itself and a small number of fingers pointing to nodes that are far from it. This configuration is claimed to provide a small number of hops to go from any node and reach any other one in the system. Such characteristics suggest that these networks could be natural candidates for the distribution of identifiers generated by LSH functions.

### 7.1 Building a Small Word based P2P network

A P2P network is a set of nodes that provides access to a set of contents through a mapping function to a virtual identifier space. This function allows the establishment of distance relationships between nodes and contents. In a Small World network, "common knowledge" can be translated into an aggregation in which contents that have similar characteristics are stored near each other by this distance metric.

To allow access to the nodes and their contents, it is necessary to embed a graph in the identifier space. This graph helps the routing service, facilitating the retrieval of a content associated with an identifier in the P2P network. The routing strategy influences the number of hops needed to retrieve a content.

Girdzijauskas [20] proposed an algorithm to build a Small World network in which the virtual identifier space is organized in a ring, and each node has local fingers with their adjacent neighbors on the right and left side of the ring; and it must have $\log_2 N$ fingers, in which N represents the estimated number of nodes in the network. This algorithm was implemented in a prototype to enable the Conceptual Search.

The P2P network generated from this graph has a binary index space distributed in a ring, with routing clockwise, a balanced distribution of the peers and an unbalanced distribution of the contents identifiers due to the semantic aggregation. In order to accomplish the prototype implementation, the virtual identifier space was divided into logarithmic

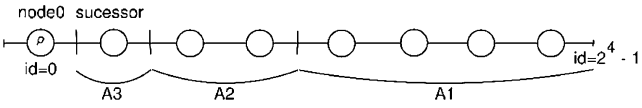


**Fig. 4** Example of virtual space with eight nodes, four bits and three partitions

partitions, as presented in Fig. 4, and making the identifiers that are meant to belong to the closest partitions of a node to be present in a larger number in its routing table. The association between a node and the identifiers for which it is responsible is similar to Chord. The nodes are organized in a virtual ring, in which each node is responsible for the storage of the keys with values between their own identifier and the identifier of their predecessor in the ring.

The whole procedure is perfectly suited for a node with identifier 0, i.e., located at the beginning of the virtual identifier space. In case of any other node we just use the identifier drawn in the preceding paragraph as an offset to be added or subtracted to the identifier of the node. This takes into account the proximity to the right (clockwise) and left (counterclockwise) in the virtual identifier space.

### 7.2 Comparing a Small World based DHT to a Chord DHT

In Chord, while there is a closeness metric between identifiers in the counterclockwise direction of the virtual space, this proximity is not important for the choice of fingers to be added in the routing table. The establishment of the fingers in Chord is done using an algorithm that considers only the ring size and distance in a clockwise direction, while in the Small World based network the establishment of the fingers is done by taking into account the proximity in the virtual space in both directions. The closer they are, the greater the chance of two nodes to establish fingers with each other is.

The next section presents some tests which were done, and we discuss the results obtained. The tests show the behavior of the aggregation of the identifiers generated by the LSH function in the virtual space of the DHT and the improvements obtained with this approach.

## 8 Evaluation

The tests done show the aggregation and the gains obtained by the use of the concept of similarity of an ontology in LSH functions, and indexing these concepts in a DHT. In the tests, keys of 128 bits were used and the ontology is shown in Fig. 2. Due to the lack of literature that addresses the indexing of multimedia data using the conceptual classification together with LSH functions, this paper does not contain any comparative test.

The first test consisted of varying the value of $m$ (Sect. 6.1) and seeing the impact of it in the aggregation of the

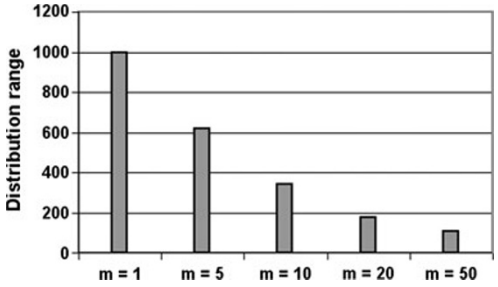


**Fig. 5** Impact of $m$ in the range of distribution

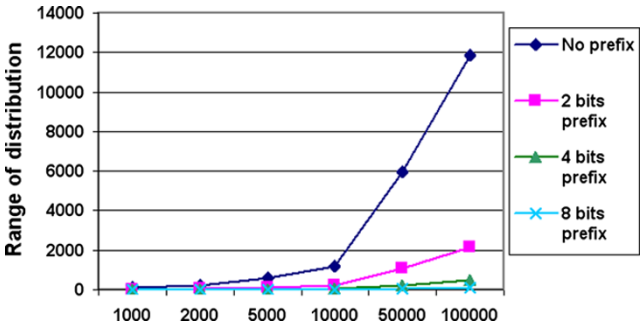


**Fig. 6** Variation in the DHT size

keys. Figure 5 shows, for $m$ assuming the values 1, 5, 10, 20 and 50, the range of the distribution of the identifiers in the virtual ring space. The space was divided into 2000 equal parts, each part representing a node in a DHT. The range is calculated by counting the number of nodes that lie between the first and the last ones that hold keys in the virtual space. In this test, the prefix showed in Sect. 6.2 was not used, as the intention was to show the impact of the aggregation due to the variation of $m$.

Figure 5 shows the average of 10 tests in each column. For larger values of $m$, the smaller is the range of distribution of the keys, in other words, the keys are stored closer to each other. This shows that $m$ is a factor that can be adjusted according to the needs of the system, if a smaller or bigger aggregation of the keys is necessary. For $m$ assuming the values 1, 5, 10, 20 and 50, the 95% confidence interval are, respectively: 180.48, 188.53, 95.51, 64.52 and 36.94.

Figure 6 shows the relation of the aggregation of the keys to the number of nodes existing in the DHT and to the variation of the size of the prefix in the keys. For these tests $m$ was fixed at 20.

As the prefix becomes larger, the more aggregated the keys that are stored and, as the size of the DHT increases, more nodes store the keys of the same ontology. For the sake of comparison, the same test was done with a traditional hash function (MD5). This type of function has linear growth, which means that the keys occupy the whole space of the ring. Because of scale issues, this result is not in the graph.

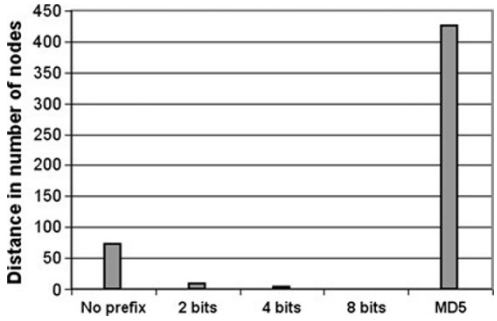


**Fig. 7** Distance from concept *Free*

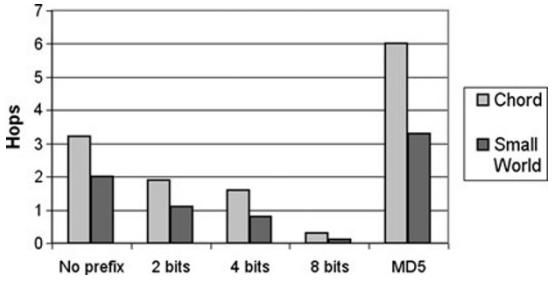


**Fig. 8** Average of the number of hops needed to reach all concepts starting from *Free*

The previous result shows that the prefix acts in the aggregation in the following way: as more bits are used in the prefix, the virtual space of the DHT is divided into a larger number of parts for the indexing of several ontologies. For example, a prefix of one bit divides the virtual space into two parts (the first part for keys with prefix equal to "0" and the second part for keys with prefix equal "1"). A prefix of two bits divides the space into four parts (identifiers with prefix equal to "00", "01", "10" and "11"), and so on. As the prefix used in the concept keys increases in size, the smaller is the virtual space in which the keys of this ontology are indexed. The number of nodes in each part varies according to the distribution and the quantity of nodes in the DHT.

If at first the results obtained with the LSH function may indicate an unbalancing in the virtual space of the DHT, the proposed system may index several ontologies that define several domains of knowledge. As shown in Fig. 3, the insertion of several ontologies in the system results in filling the ring with regions that the ontologies will occupy.

Figure 7 shows the average distance between the identifier of one concept to all other concepts, varying the size of the prefix. The number of the nodes in the virtual space was 2000 and the value of $m$ equal to 20. The concept *Free* was chosen in this test. These values are also compared with the same test using MD5.

The larger is the prefix, the closer the keys of the concepts are stored. The average for eight bits of prefix or more is almost 0, which means that actually all the concepts are stored in the same node of the DHT. Considering the keys generated by the traditional hash function, the distance between concepts is larger, resulting in a higher balance in the DHT, although in this situation there is a higher cost in the search for related concepts, as shown in the next test. In Fig. 7, the following 95% confidence interval, for the tests "No prefix", "two bits", "four bits", "eight bits" and "MD5" are, respectively: 6.09, 2.41, 0.3, and 295.94.

Figure 8 presents the number of hops necessary, starting from the node that holds the identifier of one concept, to reach all other concepts of the ontology. Again, the concept *Free* was chosen and the size of the prefix was varied. In this test, the P2P network simulator Oversim[8] was used, creating a ring with 2000 nodes in the DHT. This test was done using two different DHTs: Chord and another one, based on the Small World phenomenon [20]. The difference between the two lies in the creation of the fingers between the nodes. In Chord, the establishment of the fingers is done without giving preference to any node, while in the DHT based on Small World the creation of fingers to nodes that are near in the virtual space is privileged.

Due to the routing of Chord being done in the clockwise way, for more fairness in the tests, the identifiers were sorted and searched in ascending order. In other words, for any concept identifier $C_k$, if the concept identifier value of *Free* is smaller than $C_k$, the hops needed to go from *Free* to $C_k$ are accounted. Otherwise, the hops from $C_k$ to concept *Free* are accounted.

As seen in Fig. 8, in both DHTs, the number of hops needed to reach all other concepts keys from the key of concept *Free* is smaller if compared with the number of hops needed to the same test, but using identifiers created using MD5. The 95% confidence intervals are, to Chord "No Prefix", "two bits of Prefix", "four bits of Prefix", "eight bits of Prefix", and "MD5", respectively: 1.77, 1.03, 0.89, 0.59, and 0.86. For the tests using the Small World DHT with "No Prefix", "two bits of Prefix", "four bits of Prefix", "eight bits of Prefix", and "MD5", the 95% confidence intervals are, respectively: 0.8, 0.7, 0.44, 0.2, and 0.93.

### 8.1 Discussion about scenarios

This section discusses how the proposal presented in this paper may be applied in a real scenario of data distribution, considering the use of some Semantic Web standards.

As presented in this paper, data repositories are indexed in a P2P network using their semantic classification. This classification is based on an ontology that defines a knowledge domain. A common language to define ontologies is the OWL, a language based on RDF applied to define and

[8] Oversim: http://www.oversim.org (accessed in July 1, 2011).

```
<owl:Class rdf:about=
"http://www.example.org/SomeDomain/Music">
<rdfs:label>Music</rdfs:label>
<rdfs:comment>Root concept.</rdfs:comment>
</owl:Class>
...
<owl:Class rdf:about=
"http://www.example.org/SomeDomain/Jazz">
<rdfs:subClassOf rdf:resource=
"http://www.example.org/SomeDomain/Music"/>
<rdfs:label>Jazz</rdfs:label>
<rdfs:comment>A music genre.</rdfs:comment>
</owl:Class>
...
<owl:Class rdf:about=
"http://www.example.org/SomeDomain/Free">
<rdfs:subClassOf rdf:resource=
"http://www.example.org/AlgumDominio/Jazz"/>
<rdfs:label>Free</rdfs:label>
<rdfs:comment>
A music genre derived from Jazz.
</rdfs:comment>
</owl:Class>
```

**Fig. 9** Excerpts of an example of an OWL file

instantiate concepts of an ontology. OWL allows the description of concepts and subconcepts of an ontology, and the classification of contents in these concepts. Figure 9 shows in an OWL file an example of how to define the concepts and subconcepts of the ontology depicted in Fig. 2, using the tags *owl:Class* and *owl:subClassOf*. In Fig. 9 the concepts *Music*, *Jazz* and *Free* are declared, indicating that *Jazz* is a subconcept of *Music* and *Free* is a subconcept of *Jazz*. The Semantic Web standards recommend that each of the described data should provide an URI (Uniform Resource Identifier) indicating where more information about it can be obtained. For example, Fig. 9 indicates that more information about the concept *Free* is at http://www.example.org/SomeDomain/Free.

The classification of some data into a concept of an ontology may be expressed in their metadata, commonly written in RDF. In the metadata, it is only necessary that there exist a triple indicating this classification using the *rdf:type*[9] property. Figure 10 shows an example of classification, in which "someData", described at URI http://www.example.org/someData, is classified into concept *Jazz* of the ontology.

The topology and vocabulary of the ontology can be obtained from its description in the OWL file. These characteristics can be used to generate hash values for each concept as previously presented in this paper (Sect. 6). The classification of the data is used by a repository to take into account the conceptual value of contents it owns. Therefore, the semantic classification of a repository is defined.

[9] An RDF file may be interpreted as a collection of "subject, predicate, object" triples.

```
<rdf:Description rdf:about=
"http://www.example.org/someData">
<rdf:type rdf:resource=
"http://www.example.org/SomeDomain/Jazz"/>
</rdf:Description>
```

**Fig. 10** Excerpt of an example of RDF metadata of some data classified as *Jazz*

Even data from virtual communities or social networks (like Youtube[10], Flickr[11] and several others) can be indexed as presented in this paper. For example, users may create or own data that may be classified into some topic in a virtual community and each topic may correspond to a concept of an ontology. This classification is expressed in the metadata of the data as presented in Fig. 10. Generally, virtual communities have several users and a large volume of data and some of these data may share the classification in a knowledge domain. Using the approach described in this paper, the data from these communities would be indexed into regions of the indexing space of a P2P network, keeping their similarities. Data classified into similar concepts are stored close to each other, facilitating their retrieval in a distributed system.

For example, repositories of audio files classified as *Jazz* are indexed together in the P2P network and near other repositories that have data classified into similar concepts, like *Music*, *Free* and etc. A search for data related to *Music* and other similar concepts retrieves a list of repositories of data classified into these concepts. In the P2P network this retrieval is facilitated because of the aggregation presented in Sect. 8. Figure 11 illustrates this scenario.

## 9 Conclusion

This paper presented a proposal for the organization of conceptually classified data repositories, aiming to facilitate the conceptual search. The use of LSH functions together with the similarity between concepts of a simple ontology to create concept keys keeping this similarity was proposed. These identifiers were indexed in Chord and in a Small World based DHT and the gains in the retrieval of similar concepts were presented. These gains indicate that, using such keys, the retrieval of similar contents stored in such an indexing system demands less effort in the network, increasing the usability and feasibility of the conceptual search.

For future work, an improvement of the proposal for new scenarios and research of other ways to create the identifiers, including new ways to relate similar ontologies, is intended.

[10] Youtube: www.youtube.com (accessed in July 12, 2011).

[11] Flickr: www.flickr.com (accessed in July 12, 2011).

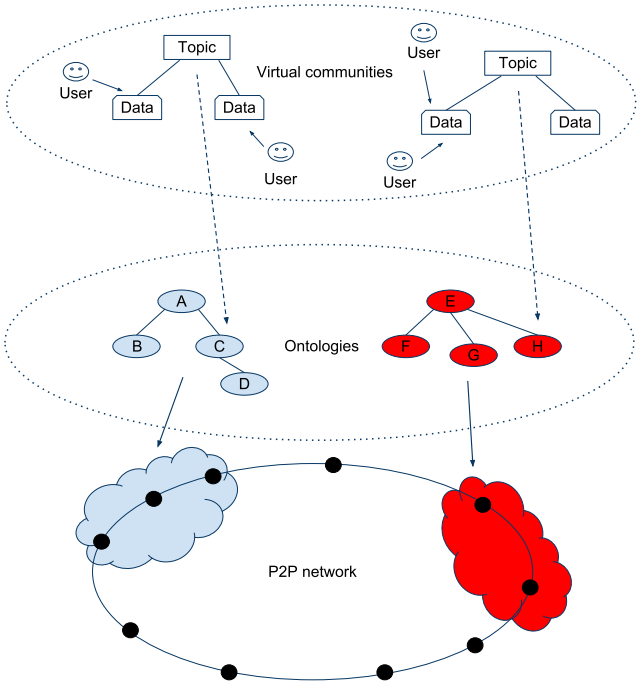


**Fig. 11** Indexing of data from virtual communities

Also the research of new ways of organization of P2P networks to facilitate even more the storage and retrieval of related data is planned. Finally, the exploration of indexing systems that use the Hamming distance as a spatial metric, as in a hypercube, is going to be considered.

## References


1. Berners-Lee T, Hendler J, Lassila O (2001) The semantic web: a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. Sci Am 284(5):29–37
2. Batista CECF, Schwabe D (2009) LinkedTube: semantic information on web media objects. In: Proceedings of the XV Brazilian symposium on multimedia and the web (Webmedia 2009)
3. Haghani P, Michel S, Cudré-Mauroux P, Aberer K (2008) LSH at large—distributed KNN search in high dimensions. In: WebDB
4. Kulis B, Jain P, Grauman K (2009) Fast similarity search for learned metrics. IEEE Trans Pattern Anal Mach Intell 31:2143–2157
5. Zhu Y (2005) Enhancing search performance in peer-to-peer networks. PhD thesis, University of Cincinnati
6. Köhler J, Philippi S, Specht M, Rüegg A (2006) Ontology based text indexing and querying for the semantic web. Elsevier Science, Amsterdam
7. Ratinov L, Roth D, Srikumar V (2008) Conceptual search and text categorization. Technical report UIUCDCS-R-2008-2932, UIUC, CS dept
8. Lyte V, Jones S, Ananiadou S, Kerr L, (2009) UK institutional repository search: innovation and discovery, Ariadne, Issue 61
9. Formica A (2008) Concept similarity in formal concept analysis: an information content approach, know-based syst. Elsevier Science, Amsterdam
10. Cordi V, Lombardi P, Martelli M, Mascardi V (2005) An ontology-based similarity between sets of concepts. In: WOA'05
11. Sridevi UK, Nagaveni N (2010) Ontology based similarity measure in document ranking. Int J Comput Appl 1(26):125–129
12. Polyvyanyy A (2007) Evaluation of a novel information retrieval model: eTVSM. Hasso Plattner Institut, Master's thesis, Univeristat Potsdm
13. Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H (2001) Chord: a scalable peer-to-peer lookup service for internet applications. In: Proceedings of ACM SIGCOMM'01
14. Indyk P, Motwani R (1998) Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the thirtieth annual ACM symposium on theory of computing (STOC '98), New York, NY, USA
15. Charikar MS (2002) Similarity estimation techniques from rounding algorithms. In: Proceedings of the thiry-fourth annual ACM symposium on theory of computing (STOC '02). ACM, New York
16. Varelas G, Voutsakis E, Raftopoulou P, Petrakis EGM, Milios EE (2005) Semantic similarity methods in wordNet and their application to information retrieval on the web. In: Proceedings of the 7th annual ACM international workshop on web information and data management (WIDM '05)
17. Gupta A, Agrawal D, El Abbadi A (2002) Approximate range selection queries in peer-to-peer systems. In: CIDR
18. Watts DJ, Strogatz SH (1998) Collective dynamics of small world networks. Nature **393**(6684), June
19. Kleinberg J (2000) The small-world phenomenon: an algorithm perspective. In: Proceedings of the thirty-second annual ACM symposium on theory of computing (STOC '00), New York, NY, USA
20. Girdzijauskas S (2009) Designing peer-to-peer overlays: a small-world perspective. PhD dissertation, Ecole Polytechnique Federale de Lausanne (EPFL), Lausane, CH, March 2009
21. Chum O, Perdoch M, Matas J (2009) Geometric min-hashing: finding a (thick) needle in a haystack
22. Georgoulas K, Kotidis Y (2010) Random hyperplane projection using derived dimensions. In: Proceedings of the ninth ACM international workshop on data engineering for wireless and mobile access (MobiDE '10)
23. Ryyndnen M, Klapuri A (2008) Query by humming of midi and audio using locality sensitive hashing. In: IEEE international conference on acoustics, speech and signal processing (ICASSP 2008)
24. Yoshida K, Murabayashi N (2008) Tiny LSH for content-based copied video detection. In: International symposium on applications and the internet (SAINT 2008)
25. Bulskov H, Andreasen T (2002) On measuring similarity for conceptual querying. In: Proc. of the 5th international conference on flexible query answering systems. Springer, Berlin
26. Chu S, Cesnik B (2001) Knowledge representation and retrieval using conceptual graphs and free text document self-organisation techniques. Int J Med Inform 62(2–3):121–133
27. Dick JP (1991) Representation of legal text for conceptual retrieval. In: Proceedings of the 3rd international conference on artificial intelligence and law (ICAIL '91)
28. Ounis I, Pasca M (1998) Modeling, indexing and retrieving images using conceptual graphs. In: DEXA '98
29. Mishne G, De Rijke M (2004) Source code retrieval using conceptual similarity. In: Proc conf computer assisted information retrieval (RIAO 04)
30. D'Amato C, Staab S, Fanizzi N (2008) 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)
31. Yang G, Oh J (1993) 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)

32. Zhu Y, Hu Y (2007) Efficient semantic search on DHT overlays. J Parallel Distrib Comput 67(5):604–616
33. Crespo A, Garcia-Molina H (2004) Semantic overlay networks for p2p systems. In: Third international workshop on agents and peer-to-peer computing (AP2PC)
34. Haase P, Siebes R, Van Harmelen F (2004) Peer selection in peer-to-peer networks with semantic topologies. In: International conference on semantics of a networked world: semantics for grid databases
35. Schlosser M, Sintek M, Decker S, Nejdl W (2002) HyperCuP—hypercubes, ontologies and efficient search on P2P networks. In: International workshop on agents and peer-to-peer computing