An Approach to Manage Semantic Heterogeneity in Unstructured P2P Information Retrieval Systems

Thomas Cerqueus, Sylvie Cazalens, Philippe Lamarre

An Approach to Manage Semantic Heterogeneity in Unstructured P2P Information Retrieval Systems

Thomas Cerqueus
LINA, UMR 6241
University of Nantes, France
thomas.cerqueus@univ-nantes.fr

Sylvie Cazalens
INRIA Sophia - LINA, UMR 6241
University of Nantes, France
sylvie.cazalens@univ-nantes.fr

Philippe Lamarre
INSA-Lyon, LIRIS, UMR 5205, F69621
University of Lyon - CNRS, France
philippe.lamarre@liris.cnrs.fr

***Abstract*—In unstructured information retrieval P2P systems, semantic heterogeneity comes from the use of different ontologies. Semantic interoperability refers to the ability of peers to communicate with each others. We take into account these notions separately, as raising two different problems. Hence we propose two independent and complementary solutions. The GoOD-TA protocol aims at reducing heterogeneity through ontology-driven topology adaptation. DiQuESh is a top-k algorithm for distributed information retrieval that is intended to ensure interoperability. This distinction enables highlighting their respective benefits on the IR performances and leads to a modular architecture. For our experiments we obtained a set of actively used real-world ontologies through the NCBO BioPortal. We implemented GoOD-TA and DiQuESH in Java and used the PeerSim simulator. We first show that GoOD-TA nicely reduces the semantic heterogeneity related to the system topology, handles the evolution of peers' descriptors, and is suitable for dynamic systems. Then, GoOD-TA and DiQuESh are run simultaneously, with a significant increase of precision and recall. This enables to identify the indirect contribution of heterogeneity reduction obtained with GoOD-TA to improving interoperability.**

I. Introduction

Peer-to-Peer (P2P) systems have proved useful for sharing resources at large scale. In addition to their scalability and dynamicity properties, they enable the peers' autonomy and decentralized control. We focus on unstructured P2P Information Retrieval (IR) systems where each peer may be viewed as both a query initiator and an autonomous provider that manages its own documents. Without reference to a centralized entity, it freely decides which documents it shares, with who it does so, and how it characterizes its documents, via some annotation or indexing process.

We target distributed information sharing applications for scientific communities. The participants may include scientists from different universities and different domains (biology, medicine, chemistry, sports, ...) but also many other actors from different companies or government institutions. Because the participants may have different objectives, contexts, viewpoints or expertise levels it is quite unlikely that they model the application domain in the same way. In addition they are free to choose whatever model that best fit their needs. This is why we assume that each peer uses an ontology (either a reference ontology or a specifically designed one). An ontology is a controlled vocabulary that models a domain in terms of *entities*, namely concepts, concept properties and relations between concepts. The entities of an ontology may be used to annotate a peers' documents. This is a simple way to provide a uniform characterization of pieces of information with very different formats such as photos, texts or experimental datasets. For example, in the biomedical domain, it is common practice to annotate scientific articles with concepts of the Medical Subject Headings (MeSH) or genetics-related datasets with concepts from the Gene Ontology. The use of different ontologies leads to a context we denote as *semantic heterogeneity*. This clearly results in a lack of *interoperability* due to the disability of some peers to precisely understand other peers' queries, thus providing marginally relevant documents.

The guideline of this paper is to clearly distinguish the semantic heterogeneity of the system, that comes from the use of different ontologies, and the semantic interoperability that refers to the peers ability to communicate with each others within the system. The former may be seen as a characterization of the difficulty to solve the interoperability problem. Indeed, if heterogeneity is low, ensuring interoperability is easier. Also, in IR, effectiveness of interoperability is evaluated using metrics such as precision, recall or F-measure, while heterogeneity better reflects a state of the system at some time point, that might evolve during the system lifetime. Considering heterogeneity reduction and interoperability improvement as independent and complementary problems leads us to consider two classes of solutions: the algorithms that impact the heterogeneity reduction, thus indirectly contributing to the interoperability increase and those which directly improve interoperability. Making this simple distinction enables to better highlight the problem, the behavior of proposed algorithms, and their respective contributions to the IR performances. The objective of this paper is then threefold. First, we aim at defining a method that reduces semantic heterogeneity. Second, we want to define a simple IR process for distributed heterogeneous context. Finally, we want to evaluate their respective contribution in improving interoperability.

There are several ways to lower semantic heterogeneity as for example facilitating the emergence of an ontology shared by peers [21][17], or making peers learn or guess more correspondences between ontology entities as in [1][9][17][6]. A good knowledge of correspondences by the peers is obviously necessary. However, we believe that in very large scale distributed systems, the most important is each peer's

neighborhood because search algorithms often restrict search around the peers. Hence, it is vital for peers to have neighbors that understand them. We focus on dynamic overlays obtained using gossiping algorithms. They enable to adapt the topology of peers without explicit semantic description of the target topology (e.g. [16]). Peers regularly exchange information about peers' descriptors and choose similar peers as neighbors After several exchange cycles, the system topology evolves. This idea has been for example used to create overlays where close peers share the same interests [4]. This solution seems well adapted to dynamic contexts, where peers join or leave the system. However, to our knowledge, no work has studied the use of gossiping to adapt the system topology according to the ontologies used by peers and the correspondences they know, in order to lower (topology-based) semantic heterogeneity.

In this paper, we bring the following contributions. First, we define the GoOD-TA protocol (Gossip-based Ontology-Driven Topology Adaptation) in section III. It aims at lowering semantic heterogeneity by organizing unstructured P2P systems according to a semantic proximity between peers descriptors. A descriptor corresponds to the synthetic description of the semantic knowledge of a peer. We propose two versions of the protocol. In the first version, the descriptors and the proximity only take into account the ontologies used by peers. The second version in addition considers correspondences that peers know. Hence the neighbors of a given peer are more likely to understand the query it issues either because they use the same ontology or because they are able to translate it. We bring solutions to manage the system dynamicity and the evolution of peers' semantic descriptors.

Second, we propose DiQuESH (Distributed Query Evaluation in Semantically Heterogeneous context) in section IV. It is a distributed top-k query evaluation protocol that aims at ensuring interoperability by enabling query translation based on ontology alignments known by peers. Contrary to many data integration solutions such as [12][1][5], the original query is forwarded in order to let each peer master the query translation on its own. The local document relevance evaluation considers the deviation of the translated query with respect to (w.r.t.) the original one. This enables to lower the effects of an approximate translation.

Third, in section V, we provide several experiments with the overall objective of studying the contribution of the GoOD-TA protocol to the IR performances of the DiQuESH process. We used the PeerSim simulator [20] to simulate the network. We used the ontologies and services of the NCBO BioPortal and research articles from the PubMed database. The advantage is that the 149 ontologies of our evaluation are actively used by the biomedical community and it is possible the get the document indexing from the BioPortal. We first provide extensive experiments showing the good behavior of the GoOD-TA protocol: it significantly reduces the semantic heterogeneity, nicely handles dynamicity and the evolution of peers' descriptors. Then, we run together the GoOD-TA and DiQuESH protocols. We show that the former significantly improves the precision and recall of the latter, thus indirectly contributing to semantic interoperability.

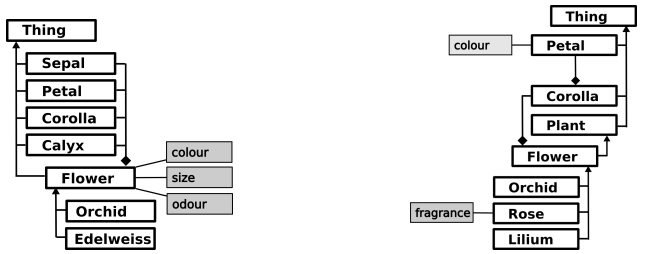

Fig. 1. Two ontologies o_1 and o_2 composed of concepts, concept properties and relations.

Comparison with related work is detailed in section VI. In our view, this paper benefits from previous contributions such as [14][2] to provide a new original approach to build a totally decentralized, two layered solution for semantically heterogeneous P2P systems for distributed information retrieval where peers annotate their documents with an ontology. A main characteristic is the total independence of the two software layers. Indeed, the way structural heterogeneity is decreased is totally disconnected from the way documents are retrieved. Together with the definition of gossip, this is a major difference with the work described in [1].

II. Model and Definitions

A. Ontologies and Alignments

An ontology provides a controlled vocabulary to model a domain. Its expressiveness varies depending on the type of ontology used. Here, we consider that an ontology is composed of *entities*, namely, a set of concepts, a set of relations and a set of properties assigned to concepts. We do not consider the instances of the concepts. The left hand side of Fig. 1 shows a part of an ontology about plants. In our experiments, we use the standard language OWL for representing ontologies, defined by the World Wide Web Consortium (W3C) [3] whereas the algorithms are presented independently of any ontology representation language. We assume that each ontology is uniquely identified by an URI.

An alignment process aims at identifying correspondences between the entities of two different ontologies [10].

Definition 1 (Correspondence): A correspondence between two ontologies o and o' is a 4-tuple $\langle e, e', r, n\rangle$ such that e is an entity from o, e' is an entity from o', $r \in \{\sqsubseteq, \equiv, \sqsupseteq, \perp\}$ is a relation between e and e' and $n \in [0, 1]$ is a confidence value representing how much the correspondence is trustworthy.
The symbols $\sqsubseteq$, $\equiv$, $\sqsupseteq$ and $\perp$ respectively stand for *less general than*, *equivalent to*, *more general than*, and *disjoint from*.

B. Unstructured P2P Systems

We model an unstructured peer-to-peer system $\mathcal{S}$ as a set of peers $\mathcal{P}$ (or nodes) connected together through a relation $\mathcal{N} \subseteq \mathcal{P} \times \mathcal{P}$. Each peer p has a unique identifier, denoted by $id(p)$, sufficient to contact it. For instance, p's identifier may be made of its IP address and a port number. To ensure relationships with other peers, peer p maintains a partial view

of the system, also called local view or view. Each entry of the view corresponds to a peer's descriptor, denoted by λ. It is at least composed of the identifier of the peer. As the number of entries may be important, only n peers are selected to be a peer's neighbors: this set of peers is noted $\mathcal{N}_p$.

When a peer joins the system, it accesses a service that enables finding neighbors. This service is always available during its time-life in the system. The *peer sampling service* provides this kind of service by returning a random sample from the set of peers [15]. Once the peer has found some neighbors, it initiates an exchange to obtain their descriptors: its view is properly initialized with its neighbors' descriptors.

C. Peers' knowledge

We model the use of an ontology by a peer by a mapping from the set of peers to a set of ontologies. The used ontology might be some reference ontology, a personalized extract of a reference ontology or a specifically designed ontology. Of course, some peers may use the same ontology.

Definition 2 (Peer-to-ontology mapping): Given a set of peers $\mathcal{P}$ and a set of ontologies $\mathcal{O}$, a peer-to-ontology mapping is a function $\mu : \mathcal{P} \rightarrow \mathcal{O}$, mapping each peer to one ontology. Peers need to know correspondences to translate (at least partially) the incoming queries and answer them. We model a peer's knowledge of correspondences by a mapping from the set of peers to the power set of a set of correspondences.

Definition 3 (Peer-to-correspondences mapping): Given a set of peers $\mathcal{P}$ and a set of correspondences $\mathcal{C}$, a peer-to-correspondences mapping is a function $\kappa : \mathcal{P} \rightarrow 2^{\mathcal{C}}$, mapping each peer to a set of correspondences.

Each peer is responsible for the evolution of its ontology and its set of known correspondences and chooses the way it manages it. We refer to existing solutions such as [13][10][21][6].

D. Information Retrieval model

In Information Retrieval (IR), the most widely used model is probably the vector space model [24]. Queries and documents are represented by a vector of weighed terms, where the terms are the space dimensions. Each term is weighted according to its representativeness of the document (resp. query). We use a variant of this model, considering concepts instead of terms. For example, considering the vector space defined by ontology o_2 (cf. Fig. 1), a document vector could be: $[(Element, 0), (Petal, 0.8), (Corol, 0), (Calix, 0), (Vegetal, 0), (Flower, 1), (Orchidea, 0), (Rose, 0), (Lys, 0.3)]$. The semantic vector of a given document or query may be obtained by automatic indexing or manual annotation. We do not assume any specific type of documents. They may be textual or multimedia documents, web pages, medical diagnoses, etc. We denote by $\overrightarrow{d_o}$ the semantic vector defined over ontology o and representing document d. In a same space (same ontology), its relevance score w.r.t. a query $\overrightarrow{q_o}$ is given by the cosine similarity, which is generally used in IR vector spaces:

$$\cos(\overrightarrow{d_o}, \overrightarrow{q_o}) = \frac{\overrightarrow{d_o} \cdot \overrightarrow{q_o}}{|\overrightarrow{d_o}| \times |\overrightarrow{q_o}|}$$

III. Topology Adaptation Protocol

Basically, in an information retrieval process, peers send queries in their neighborhood because flooding the whole system is unrealistic. In this context it is crucial to ensure that peers receiving queries issued by a peer understand this peer. In order to do that, we propose the GoOD-TA protocol.

A. Principles of the GoOD-TA protocol

GoOD-TA is a gossip-based protocol [18] that makes peers exchange descriptors of other peers, so that the topology of the system evolves. Each peer consists of two threads: an active and a passive one. The active thread is used to initiate communication with another peer: Each peer p regularly (each θ time units) contacts another peer to exchange descriptors of other peers. When peer p' is contacted by p (through the passive thread), p' has to answer by returning a list of descriptors. Then, both peers treat the received descriptors: they use them to build their partial view of the system. Notice that each peer has to define its own descriptor, changing it when its knowledge changes. It is the only one entitled to do so (except in one case, cf. section III-E). We assume that peers are not malicious: each peer provides a correct descriptor of itself. Peers have to process three crucial tasks: peer selection, data selection and data processing. We describe them w.r.t. some peer p.

Peer selection: The peer selection is done by randomly selecting a peer p' in the local view. Thus the peer p executing the selection knows the descriptor of p'. If the view does not contain descriptors enough to discover new neighbors, the *peer sampling service* [15] can be invoked to refresh/reset the view.

Data selection: When peer p has to send data to p', it decreasingly ranks the descriptors of its own view according to their proximity with p'. Then p sends the best (closest) descriptors to p'. Ranking the view w.r.t. p' allows to reduce the convergence speed of the protocol. Only m_{max} descriptors are considered in order not to overload the network.

Data processing: When p receives a set of descriptors from p', it merges its view with the received descriptors. The n closest peers of the view (those which are the most similar to p) become its neighbors. A descriptor is not added in the view if it is already in it. If the storage space of p is limited, then only v_{max} entries are kept.

In order to rank the descriptors in the peers' views, the data selection and data processing phases use a proximity function. We define it as a function $prox_\lambda(\lambda')$ returning a value in $[0, 1]$ that corresponds to the proximity of a descriptor λ' w.r.t. another descriptor λ. If $prox_\lambda(\lambda')$ equals 1, then p' understands all the concepts of the ontology of p. A proximity function verifies the following property: $\lambda = \lambda' \Rightarrow prox_\lambda(\lambda') = 1$. The inverse is not necessarily true: the proximity may equal 1 even with different descriptors.

We present two approaches. For each of them, we define the content of a peer's descriptor and an associated proximity function.

B. Light Version

A peer p is described by its identifier (id) and the identifier of its ontology $\mu(p)$ noted $onto\_id$. Table I is an example of a peer's view. We define the proximity function by:

$$prox_\lambda(\lambda') = \begin{cases} 1 & \text{if } \lambda.onto\_id = \lambda'.onto\_id \\ 0 & \text{otherwise} \end{cases}$$

Considering a peer p_4 using o_2 and having the Table I as view, we find: $prox_{\lambda_4}(\lambda_2) = 0$, $prox_{\lambda_4}(\lambda_9) = 1$, $prox_{\lambda_4}(\lambda_3) = 0$, and $prox_{\lambda_4}(\lambda_7) = 1$.

TABLE I
PEERS' DESCRIPTORS IN A VIEW (LIGHT VERSION).

	id	$onto\_id$
λ_2	$id(p_2)$	$uri(o_1)$
λ_9	$id(p_9)$	$uri(o_2)$
λ_3	$id(p_3)$	$uri(o_3)$
λ_7	$id(p_7)$	$uri(o_2)$

This approach ranks peers according to the ontology they are using. Hence, peers move closer to peers that use the same ontology. This approach is very convenient because it requires very little information. Nevertheless, it could be inefficient when some ontologies are shared by small numbers of peers: it may be difficult for them to meet. Moreover, the definition of proximity does not allow to distinguish the case where two ontologies have 99% of their concepts in common, and the case where only 5% of their concepts are in common. And it does not consider the peer's knowledge of correspondences either. As a consequence, we propose a more refined approach.

C. Refined Version

Although it would enable to compute a very accurate disparity between peers, it is quite unrealistic to include the known correspondences in a peer's descriptor. Hence, we propose to consider what looks like a coarse approximation: the number of correspondences the peer knows between its ontology and others. The descriptor of a peer p is made of (i) the identifier of the peer: id, (ii) the identifier of the ontology used by the peer: $onto\_id$, (iii) the number of concepts contained in the ontology: $onto\_size$, (iv) a set $corr$ of triples $\langle o, o', nb\rangle$ where nb is the number of correspondences that p knows between ontologies o and o' (*i.e.* a subset of $\kappa(p)$). In this paper, we only consider equivalences ($\equiv$), but the proposed solution could be generalized to other types. Each triple $\langle o, o', nb\rangle$ concerns p: p uses o or o'. Table II presents a peer's view. The second line shows that peer p_9 uses o_2 (which contains 95 concepts) and knows 72 correspondences between o_2 and o_1, and 36 between o_2 and o_4.

Given this richer definition of descriptor, we intend to reflect that the proximity of λ' w.r.t. λ depends on the capacity of p' to understand concepts from the ontology of p. We define the proximity as:

$$prox_\lambda(\lambda') = \begin{cases} 1 & \text{if } \lambda.onto\_id = \lambda'.onto\_id \\ \frac{nb}{\lambda.onto\_size} & \text{if } \exists\langle \lambda.onto\_id, \lambda'.onto\_id, nb\rangle \\ & \qquad \in \lambda'.corr \\ 0 & \text{otherwise} \end{cases}$$

TABLE II
PEERS' DESCRIPTORS IN A VIEW (REFINED VERSION).

	id	$onto\_id$	$onto\_size$	$corr$
λ_2	$id(p_2)$	$uri(o_1)$	110	$\{\langle uri(o_1), uri(o_2), 85\rangle\}$
λ_9	$id(p_9)$	$uri(o_2)$	95	$\{\langle uri(o_2), uri(o_1), 72\rangle,$ $\langle uri(o_2), uri(o_4), 36\rangle\}$
λ_3	$id(p_3)$	$uri(o_3)$	1,417	$\emptyset$
λ_7	$id(p_7)$	$uri(o_2)$	95	$\{\langle uri(o_2), uri(o_3), 58\rangle\}$

This function does not satisfy the symmetry property. Indeed the fact that p_2 knows nb correspondences between $\mu(p_1)$ and $\mu(p_2)$ does not imply that p_1 also knows these correspondences. For a peer p_4 using o_2 (which contains 95 concepts) and having the Table II as view, we find that: $prox_{\lambda_4}(\lambda_2) = \frac{85}{95}$ because p_2 has 85 correspondences between o_1 and o_2; $prox_{\lambda_4}(\lambda_9) = 1$ because p_9 also uses o_2; $prox_{\lambda_4}(\lambda_3) = 0$ because p_3 has no correspondence between o_3 and o_1, etc.

D. Evolution of peers' semantic descriptors

In this section we deal with the fact that the descriptor of a peer p may change. It may happen because: (i) p decides to use another ontology (a new ontology), (ii) p makes its ontology evolve, (iii) p discovers new correspondences between its ontology and another one. To support these evolutions, we add a number version to each descriptor (v). A local clock is used to initialize the number version. For instance the number version can be an integer representing the number of seconds since January 1, 1970. The number version is updated when the peer's descriptor changes. Notice that the number version is used to compare two versions of a peer's descriptors: it is never used to compare descriptors of two different peers. When a peer p receives a set of descriptors (during the data processing), it must check if each descriptor is already in its view and keep the latest version. We illustrate the proposed solution with two scenarios.

Scenario 1: Let us consider a peer p_2 using an ontology o_2 of 95 concepts, and having 50 correspondences between o_2 and o_1. If we consider the refined version of GoOD-TA, the descriptor of p_2 is: $[id(p_2), uri(o_2), 95, \{\langle o_2, o_1, 50\rangle\}, v_1]$. After a while, p_2 discovers new correspondences. Then the descriptor is changed and the version number is updated: $[id(p_2), uri(o_2), 95, \{\langle o_2, o_1, 95\rangle\}, v_2]$. At this point p_2 realizes that it could use o_1 (containing 110 concepts) rather than o_2 because all the concepts of its ontology are mapped to concepts of o_1. Then it gets the ontology o_1 and starts to use it. This situation is realistic, in particular if o_1 and o_2 come from a common ontology. As a consequence, a new descriptor is created: $[id(p_2), uri(o_1), 110, \{\langle o_1, o_2, 95\rangle\}, v_3]$. Known correspondences are kept because they still involve p_2's ontology.

Scenario 2: We consider a peer p_2 using an ontology o_2 of 95 concepts, and knowing 55 correspondences between o_2 and o_1. Its descriptor equals: $[id(p_2), uri(o_2), 95, \{\langle o_2, o_1, 55\rangle\}, v_1]$. At this point p_2 choses to add some concepts in the ontology o_2 to fit with its need. A new ontology o'_2 is created: it contains concepts of o_2 and 6 other concepts. Then o'_2 contains 101 concepts of which 95

are in common with o_2: 95 correspondences exist between o'_2 and o_2. Besides the triple $\langle o_2, o_1, 55\rangle$ is not relevant for p_2 because it is not using o_2 anymore. But as all concepts of o_2 are mapped with those of o'_2, the triple $\langle o'_2, o_1, 55\rangle$ can be considered. Thus p_2's descriptor is:
$[id(p_2), uri(o'_2), 101, \{\langle o'_2, o_1, 55\rangle, \langle o'_2, o_2, 95\rangle\}, v_2]$.
The identifier of o'_2 ($uri(o'_2)$) must be unique. It can be computed using the identifier of o_2, the identifier of p_2 and the current value of p_2's clock. It ensures that several peers can create different ontologies from a common source at the same time, and that a peer can create different ontologies from a common source at different times.

E. Dynamicity of the system

In this section we deal with the fact that P2P systems are dynamic: peers can join or leave the system at any time. To take it into account, we introduce a new notion: the status of a descriptor. It can basically be represented by a boolean value (`0` or `1`). During its presence in the system, a peer p communicates its descriptor (through the classic protocol) with a status equal to `1`. This value is unchanged while p is in the system. If p leaves the system (voluntarily or because of a failure), its descriptor becomes obsolete because it is not reachable anymore. We distinguish between both cases.

Peer Departure: When peer p chooses to leave the system, it changes its descriptor: it puts the status to 0 and updates its version number. In the light version, peer p would turn its descriptor $[id(p), uri(o), v, 1]$ into $[id(p), uri(o), v', 0]$, where $v' > v$. Before leaving, p sends its descriptor to its neighbors. This way they learn that p left, and they are able to convey this information to other peers by sharing its new descriptor.

Peer Failure: If peer p fails, the descriptor cannot be updated by p itself. So when a peer p' observes that p is not reachable anymore, it modifies the descriptor $desc(p)$: it simply sets the status to `0` and increments the version number by one. Peers clocks are not necessarily synchronized so p' does not use its local clock to update p's descriptor. In the light version of GoOD-TA, p' would turn p's descriptor $[id(p), uri(o), v, 1]$ into $[id(p), uri(o), v', 0]$ where $v' = v + 1$. Notice that this is the only situation in which a peer is allowed to modify the descriptor of another peer. After that it continues to share this descriptor in order to inform other peers that p left. The peer failure detection is ensured by an independent mechanism which is out of the scope of this paper.

Peer Join: When a peer p joins the system, the version number is initialized using the local clock, and the status is set to `1`. If it is not the first time that p joins the system, the new version number is greater than the number version propagated in previous sessions.

IV. Distributed Query Evaluation

DiQuESH is a distributed top-k query evaluation protocol in semantically heterogeneous environment. It provides the query initiator with a set of k best results within a given neighborhood, defined by a TTL value: Considering the same neighborhood, there is no result outside this set that has a strictly better score. A result is a triple (id_p, id_d, sc_d): the peer identified by id_p assigns the relevance score sc_d to the document locally identified by id_d. Hence during execution, it is not possible to know whether a same document is returned several times by different peers. We assume that the scores are normalized in $[0, 1]$ and comparable. We adapt the generic algorithm *Fully Distributed* [2] to the case of IR in semantically heterogeneous context. We describe the different steps.

A. Query forward

A query initiator p sends the query to its direct neighbors ($\mathcal{N}_p$). The message is made of the semantic vector characterizing the search $\overrightarrow{q_o}$, expressed relative to its ontology o, the URI of o, the number k of required documents and a given TTL. When some peer p' receives a query $\overrightarrow{q_o}$ with $TTL \neq 0$, it first forwards it to its neighbors and decreases the TTL value. Then it treats it locally.

B. Local query execution

This step provides the k best local results. It has two stages, query translation and relevance scoring.

Query translation: To translate a query $\overrightarrow{q_o}$, a peer p' with ontology o', uses the correspondences it knows between o et o'. This results in a semantic vector $\overrightarrow{q_{o'}}$, expressed in the vector space defined by o'. To keep simple, this process (described in Algorithm 1) only considers the equivalences which confidence value n is equal to 1 (cf. line 3). We could as well consider those where m is above some threshold, and modulate the weight of concepts according to m. We could also consider other types of correspondences (subsumption, etc.) in order to weight additional concepts in the space defined by o', but with the well known limits of any expansion process.

Algorithm 1: Translation of vector $\overrightarrow{v_o}$ by peer p'.

Input: A semantic vector $\overrightarrow{v_o}$ expressed relative to o.
Output: A semantic vector $\overrightarrow{v_{o'}}$ expressed relative to o'.

```
1 v_o' ← ∅ // Weights initialized with 0
2 for c ∈ v_o do
3     if ∃c' ∈ C_o' : ⟨c, c', ≡, 1⟩ ∈ κ_p'(o, o') then
4         v_o'[c'] ← v_o[c] ;   // v_o[c] is the weight
          of c in v_o
```

Local relevance scoring: Peer p' could compute the document relevance considering the translated query only and forgetting the initial query. However, the risk is to assign an inappropriate relevance value. For example, let us consider an initial query represented by $[(c_1, 1), (c_2, 1), (c_3, 1)]$ with a translation represented by $[(c'_1, 1)]$ only, with $c_1 \equiv c'_1$ and a document d represented by a single concept $[(c'_1, 1)]$. Then considering the cosine similarity, the relevance value of d is equal to 1. However, another peer might know more alignments and better translate the query as: $[(c'_1, 1), (c'_2, 1), (c'_3, 1)]$. Then the relevance value of d is lower. Hence, it seems that a

relevance value that has been computed w.r.t. an approximative translation should be penalized. This is all the more important as the algorithm compares the scores from different peers. Those which perform accurate translations would be wrongly penalized.

Our proposal is to adjust the score of the document relative to the translated query by taking into account the deviation of the translated query relative to the initial query. The deviation corresponds to the "error" introduced by the incomplete query translation. Any receiving peer can compute it, even if it does not have all the query concepts in its ontology. Hence, the relevance score of a document $\overrightarrow{d_{o'}}$ relative to an initial query $\overrightarrow{q_o}$ is given by :

$$score(\overrightarrow{d_{o'}}, \overrightarrow{q_o}) = \cos(\overrightarrow{d_{o'}}, \overrightarrow{q_{o'}}) \times \cos(\overrightarrow{q_o}, \overrightarrow{\tilde{q}_o})$$

where $\overrightarrow{\tilde{q}_o}$ is the vector that corresponds to the initial query $\overrightarrow{q_o}$ limited to the concepts that have been translated and considered in $\overrightarrow{q_{o'}}$. Notice that when d and q are represented within the same space defined by o, we have $\overrightarrow{q_o} = \overrightarrow{\tilde{q}_o}$ and, then $score(\overrightarrow{d_{o'}}, \overrightarrow{q_o}) = \cos(\overrightarrow{d_{o'}}, \overrightarrow{q_{o'}})$.

This approach has two advantages. First, the document scores are comparable as they consider the deviation between the initial query and the query that is actually evaluated. Second, it is rather generic and could be used with other relevance measures.

C. Merge and backward - retrieval

After treating a query locally, peer p' waits for its neighbors' results lists. It performs a merge and sort algorithm of its own list with the received ones. Then it selects the k best results. It sends back the obtained list to the peer that had forwarded him the query. To actually get the k most relevant documents, the query initiator directly contacts the peers. This is possible because a result is a triple containing the peer's address and the document local identifier.

V. Preliminary Experiments

A. Dataset and P2P simulator

In order to run our experiments we need a set of ontologies, a set of alignments between these ontologies, a set of documents and queries indexed/annotated with concepts of these ontologies, and a relevance judgment. To our knowledge, no such predefined IR corpus is available. We used the BioPortal Web services [11] to get some of these elements but their size (e.g. the number of ontologies) limits the scope of the results presented in sections V-C and V-D.

We obtained 149 OWL ontologies that are actively used in biomedical communities. We also retrieved $1,417$ alignments between these ontologies. They contain $28,027$ correspondences ($\equiv$) between 91 ontologies: 58 ontologies are totally disconnected from others. Globally ontologies are very dissimilar: they hardly overlap. The problem of semantic heterogeneity is particularly difficult. In order to run information retrieval (IR) experiments, we downloaded semantic annotations of $4,163$ documents (through a BioPortal service). Documents correspond to articles published in biomedical journals that come from the PubMed database. Some documents are annotated w.r.t. different ontologies. Annotations are used to build semantic vectors.

We implemented GoOD-TA and DiQuESH in Java. We used the PeerSim simulator [20] to generate P2P systems as random directed graphs of peers. Each peer is linked to some others peers which are used to initialize its view. Ontologies are randomly assigned to peers according to a discrete uniform distribution. Each peer initially knows correspondences involving its own ontology. PeerSim uses a *seed* to simulate randomness. Each experiment was launched three times with different seeds. The results presented correspond to the average of these three experiments.

B. Metrics

Semantic heterogeneity: We use two metrics defined in [7]: $\mathcal{H}_{Rich}$ and $\mathcal{H}_{DapAvg}$. They focus on different facets of heterogeneity, and they are normalized in $[0, 1]$. The $\mathcal{H}_{Rich}$ measure, which characterizes the semantic richness, is defined as:

$$\mathcal{H}_{Rich}(\mathcal{S}) = \frac{|o_{\mathcal{S}}| - 1}{|\mathcal{P}| - 1}$$

where $o_{\mathcal{S}}$ is the set of ontologies used in $\mathcal{S}$, and $\mathcal{P}$ is the set of peers in $\mathcal{S}$. The richness measure gives information about semantic diversity of the system but does not take into account the organization of the system (*i.e.* the neighborhood relations between peers). The neighborhood of a peer p, noted $\mathcal{N}_p^r$, represents the set of peers accessible from p with at most r hops (p does not belong to its own neighborhood). We consider the $\mathcal{H}_{Dap}$ metric which measures the heterogeneity around a specific peer p. It is defined as:

$$\mathcal{H}_{Dap}^r(\mathcal{S}, p) = \frac{1}{|\mathcal{N}_p^r|} \sum_{p_i \in \mathcal{N}_p^r} d(p, p_i)$$

In these experiments, the disparity function d is defined as:

$$d(p, p') = \begin{cases} 0 & \text{if } o = o' \\ 1 - \frac{|\kappa_{p'}(o, o')|}{|C_o|} & \text{otherwise} \end{cases}$$

where o and o' are the ontologies used by p and p', C_o is the set of concepts of o, and $\kappa_{p'}(o, o')$ is the set of correspondences that p' knows between o and o'. This measure determines at which point p' misunderstands the concepts of p. $\mathcal{H}_{Dap}$ is used to define a global measure:

$$\mathcal{H}_{DapAvg}^r(\mathcal{S}) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \mathcal{H}_{Dap}^r(\mathcal{S}, p)$$

It allows to characterize the heterogeneity of a system according to its topology. If each peer p is surrounded by semantically close neighbors, then each $\mathcal{H}_{Dap}^r(\mathcal{S}, p)$ is close to 0. As a consequence, $\mathcal{H}_{DapAvg}^r(\mathcal{S})$ is also close to 0.

IR performances: In order to measure the efficiency of an information retrieval method we consider the precision and the recall metrics. Given a top-k query q, they are defined as:

$$P_q = \frac{|\mathcal{A}_q \cap \mathcal{R}_q|}{|\mathcal{A}_q|} \text{ and } R_q = \frac{|\mathcal{A}_q \cap \mathcal{R}_q|}{\min(k, |\mathcal{R}_q|)}$$

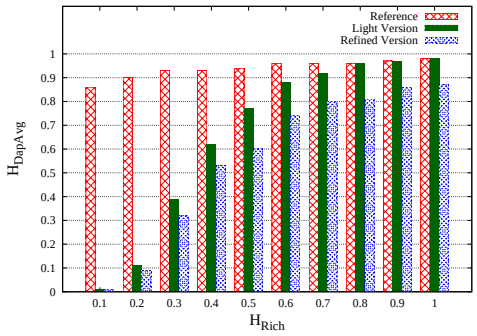

Fig. 2. Comparison of both versions of GoOD-TA in different situations of semantic diversity ($\mathcal{H}_{Rich}$).

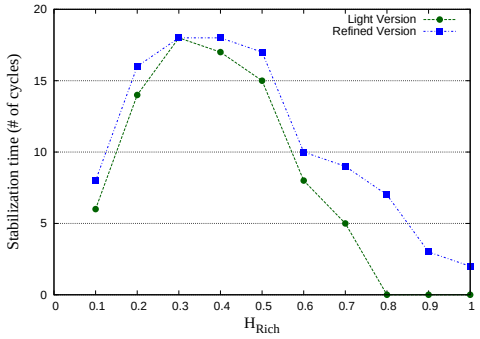

Fig. 3. Stabilization time of GoOD-TA according to the semantic richness.

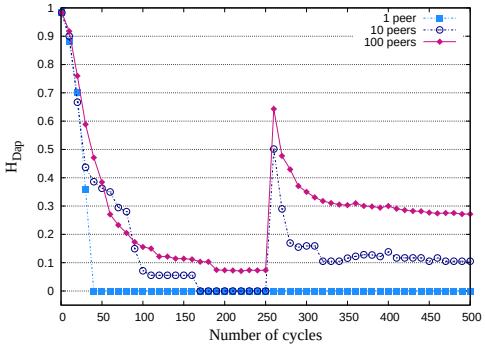

Fig. 4. Evolution of $\mathcal{H}_{Dap}$ around peers that changed their ontology.

where $\mathcal{A}_q$ is the set of retrieved documents for q, and $\mathcal{R}_q$ is the set of relevant documents for q.

C. Semantic diversity tolerance

In this section we study the performances of GoOD-TA in different situations of semantic heterogeneity: We evaluate its capacity to reduce the topology-related heterogeneity (measurable with $\mathcal{H}_{DapAvg}$) according to the semantic diversity of the system (measurable with $\mathcal{H}_{Rich}$). In this experiment, we aim to vary the diversity degree between 0 and 1. As we only have 149 ontologies, we are obliged to consider P2P systems of 149 peers. We vary the number of ontologies: 16, 31, 45, 60, 75, 90, 105, 119, 134 or 149. Thus, $\mathcal{H}_{Rich}$ equals 0.1, 0.2, ..., and 1. After the initialization step, the GoOD-TA protocol runs during 300 cycles, point where heterogeneity is stabilized. The number of descriptors sent at each cycle is limited to 5. The size of local caches is set to 20. For this experiment peers do not leave/join the system and do not change their descriptors. The degree of connectivity is set to 3: each peer is directly connected to 3 other peers. The radius r is set to 3.

The results of the experiment are shown on Fig. 2. The reference corresponds to the case where GoOD-TA is not running. We can see that both versions are very efficient when the diversity is lower than 0.8. The more diversified the system, the less efficient the protocol. When $\mathcal{H}_{Rich}$ is greater than 0.8, the light version of the protocol is ineffective. The refined version still reduces $\mathcal{H}_{DapAvg}$ when $\mathcal{H}_{Rich}$ equals 1. In all the situations, the refined version is more efficient than the light one: heterogeneity $\mathcal{H}_{DapAvg}$ is more reduced.

D. Stabilization speed

We aim to compare the stabilization speed of both versions of GoOD-TA. Notice that the stabilization is ensured while peers store at least $\log(|\mathcal{P}|)$ entries in their views [16]: it is the case in our experiments. The parameters are those described in section V-C and the simulation lasts 300 cycles. Let M be the minimal value of heterogeneity observed during the simulation: $M = \min_{t\in[0,300]} \mathcal{H}[t]$. We define the stabilization time st as the minimal time after which all the values of heterogeneity are close to M: $\forall t \in [st, 300], \mathcal{H}[t] \leqslant M + \varepsilon$ (in this experiment $\varepsilon = 0.05$).

Fig. 3 presents the convergence speed of the two versions of GoOD-TA in different situations of semantic diversity. Both curves have the same shape. The convergence times increase when the value of $\mathcal{H}_{Rich}$ is between 0 and 0.3, and they decrease when $\mathcal{H}_{Rich}$ is greater than 0.4. Convergence is rapid when $\mathcal{H}_{Rich}$ is low (it is easy to reduce $\mathcal{H}_{DapAvg}$) and when $\mathcal{H}_{Rich}$ is high ($\mathcal{H}_{DapAvg}$ is slightly reduced). Convergence is slow when $\mathcal{H}_{Rich}$ is medium (lots of descriptors exchanges are needed). The convergence speed of the light version equals 0 when $\mathcal{H}_{Rich}$ is greater than 0.8 because in this case $\mathcal{H}_{DapAvg}$ is not reduced at all (see Fig. 2).

E. Evolution of peers' semantic descriptors

In this section we study GoOD-TA when some peers change their descriptors. We consider systems of $1,000$ peers using 149 ontologies. Other parameters are those used in previous sections. In this experiment, we let GoOD-TA run during 500 cycles. During the first 250 cycles, peers do not change their ontology. At the 250^{th} cycle, a number of peers change their ontology (1, 10 or 100 peers). Their descriptors are updated accordingly. New ontologies chosen by peers are potentially already used by other peers of the system. The challenge for these peers is to find new relevant neighbors. We study the local impact of these changes for peers using the $\mathcal{H}^r_{Dap}(\mathcal{S}, p)$ measure around each peer p that changed its ontology, and the average of the values is computed. We only consider the light version of the protocol (the refined version gives similar results).

Fig. 4 shows that when some peers change their ontology, they quickly find new relevant neighbors. The proximity with the new neighborhood is not necessarily the same than with the old one. It depends on the number of peers that use the same ontology in the system. To speed up the reconfiguration of the system, it is possible to make peers restart the protocol (*i.e.* empty their views and start exchanging with some peers) as it is suggested in [16]. Obviously additional experiments have to be conducted to study the performances of GoOD-TA when peers constantly and intensively make their ontology evolve, and discover new correspondences.

F. Dynamicity of the system

In order to simulate dynamicity, *i.e.* frequent joining or leaving of peers, we remove and add a given number of peers at each cycle. This way, the size of the system always remains the same. The removal of a peer corresponds to the situation in which it fails. It is considered as a critical situation because it does not inform other peers that he is leaving. We study the performance of GoOD-TA when the session duration average varies between 1 minute and 60 minutes. The session duration of a peer corresponds to the time it remains in the system. We consider P2P systems of $1,000$ peers using 149 ontologies (as a consequence $\mathcal{H}_{Rich}$ equals 0.15). The connectivity degree is set to 4. Other parameters are those used in previous sections. Peers joining the system use ontologies that are potentially already used in the system. We considered the configurations presented in Table III with cycle length of 5 seconds. When the average session duration equals x cycles, the churn rate r equals $\frac{100}{x}\%$ ($r \times |\mathcal{P}|$ peers join/leave at each cycle). In this context a churn rate greater than 1% is a critical and unlikely case because it means that peers remain only 8 minutes in the system in average (this time does not seem sufficient to share or download data). Our experiment consists in observing a system during 300 cycles.

TABLE III
CONFIGURATIONS CONSIDERED TO STUDY GOOD-TA WITH CHURN.

Average session duration (# of cycles)	Average session duration (minutes)	Churn rate (%)
12	1	8.33
60	5	1.67
180	15	0.54
360	30	0.27
720	60	0.14
∞	∞	0 (no churn)

Fig. 5 presents the semantic heterogeneity according to the average session duration for the two versions of GoOD-TA. The reference corresponds to case where no protocol is running. The curves corresponding to both versions of GoOD-TA also coincide. It means that they have the same ability to handle dynamicity. It also shows that, with both versions of GoOD-TA, heterogeneity is reduced even if peers remain a single minute in the system. When the average session duration becomes more important, the semantic heterogeneity is more reduced. We can see that when the session duration equals 60 minutes, the heterogeneity is reduced from 0.96 to 0.2. In [25]

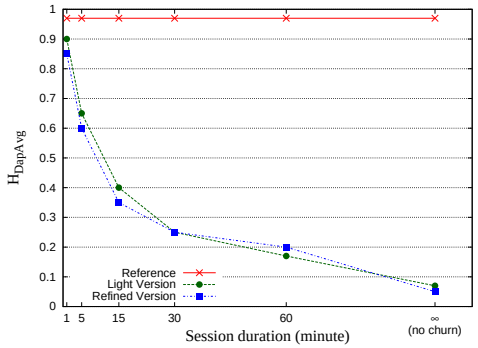

Fig. 5. Semantic heterogeneity $\mathcal{H}_{DapAvg}$ obtained after 300 cycles according to the average session duration (*i.e.* churn rate).

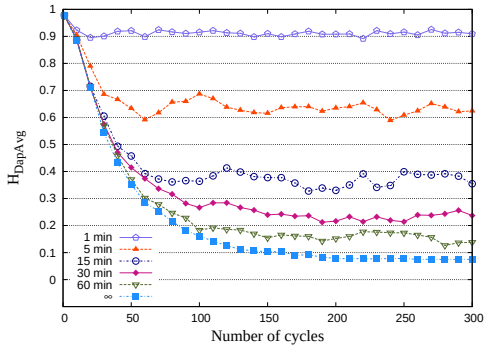

Fig. 6. Evolution of $\mathcal{H}_{DapAvg}$ during the execution of the light version of GoOD-TA according to the average session duration (*i.e.* churn rate).

authors point out that half of the participants remain in data sharing systems (e.g. Gnutella and Napster) more than one hour. Based on this analysis, we can say that GoOD-TA is suitable for these kind of systems. Fig. 6 presents the evolution of $\mathcal{H}_{DapAvg}$ for different churn rates when the light version is running. It shows that when the average session duration decreases, the heterogeneity $\mathcal{H}_{DapAvg}$ is less reduced. When the churn rate is important the system is unstable, and heterogeneity oscilates. To conclude we can say that GoOD-TA is suitable for dynamic P2P systems.

G. Improvement of semantic interoperability

Only 39 ontologies are used to represent the documents that we downloaded. We generated $1,353$ queries using concepts of these 39 ontologies. Each query contains between 1 and 3 concepts. The relevance judgment is also generated according to queries and documents. For each query q, we identified the best k ($k = 10$) documents by executing the method in a centralized system in which all the documents and all the correspondences are available. These documents form the set $\mathcal{R}_q$ of relevant documents for q.

In this experiment we consider following configuration: $1,000$ peers, 149 ontologies, no churn, no evolution of peers' semantic descriptors, connectivity degree $|\mathcal{N}_p| =$, radius $r = 3$, $TTL = 3$, and $k = 10$. As the TTL is set to 3, each peer can communicate with at most 84 ($= 4 + 4^2 + 4^3$)

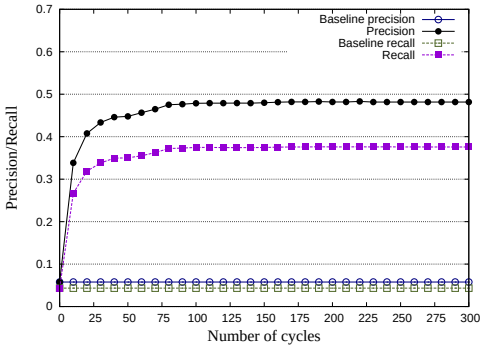

Fig. 7. Information retrieval performances of DiQuESH obtained when the light version of GoOD-TA is running.

peers. As we consider systems of $1,000$ peers, it represents 8.4% of the system. The light version of GoOD-TA runs as in previous experiments (the refined version should give better results). Every 10 cycles, randomly choosen peers issue the queries and retrieve documents through the DiQuESH algorithm. Precision and recall are then computed.

Fig. 7 presents the results of this experiment. Baselines precision and recall correspond to results obtained when no protocol is running. Fig. 7 shows that precision and recall are rapidly increased (from 0.05 at the beginning to 0.37 and 0.48 after 75 cycles). We can see that they are stable from the 100^{th} cycle whereas heterogeneity still decreases (see Fig. 6, curve ∞). This experiment shows that reducing heterogeneity allows to improve IR performances, but it also demonstrates that it is not necessary to perfectly cluster peers because the benefits w.r.t. IR performances are limited. Nevertheless it allows to easily handle dynamicity of peers.

VI. Related Work

Many works have used gossiping as presented in [18] to adapt the system topology to their needs. They differ in the way they define the peer's descriptors that are exchanged during gossip and the proximity function. For example, [16] presents general results showing how to organize P2P systems to fit target topologies such as torus. The idea of putting closer similar peers so that they can interact more easily is used in several works, which we cite only a few due to space limitations. For example, [27] consider the peers' content, without considering any metadata nor ontology while [4] consider exchanging peers' profiles, creating dynamic overlays of users with same interests but who may not know each others. However, to our knowledge, no one has focused on using this type of gossiping in order to put closer peers with similar ontologies or enough knowledge of mappings to understand their neighbors' queries.

Gossiping may also be used to reduce other facets of semantic heterogeneity. Indeed, if the peers know a lot of correspondences between entities of different ontologies, heterogeneity is lower. Knowledge of correspondences may be shared across the system by gossiping them [18]. We have studied this type of solution in [6]. It is also proposed in a different way in [21]. These approaches are complementary to the work presented in this paper and would correspond to another module in the peers' architecture.

Gossiping is not the only way to obtain overlays of semantically close peers and many works have proposed other solutions which flexibility degree varies. For instance, super-peer-based infrastructures have been proposed in [19][22], based on the clustering of peers that use the same schema. They do not correspond to our context of unstructured P2P systems and may not be very suitable when peers change their ontologies. Changes of ontologies may also be a problem in PARIS, a semantic overlay network architecture with an hybrid topology [8] where peers using identical ontologies form a local group, some of them participating to a distributed hash table to maintain a connection between the different groups. Some works focus on computing an ontology summary for each overlay (cluster ontology) [26]. Their peers clustering is mainly incremental as each incoming peer searches for the closest cluster. However, contrary to our proposal, this solution requires the explicit representation of clusters and its management during the system lifetime.

In the field of data bases, Peer Data Management Systems (PDMSs) address the management of heterogeneous schemas in P2P systems, in order to integrate structured or semi-structured data. A pioneering work is [1], where a peer's neighborhood is composed of peers with similar schema and peers against which schema it is able to translate queries. However, the way these overlays are created results from the analysis of successive query translations to identify a potential neighbor and from a simple exchange of ping/pong message to establish a real connection. This is rather far from the way we create overlays. The term "semantic gossiping" is introduced in [1] too, where it means that one "can propagate queries towards nodes for which no direct translation link exists." This is very different from our definition of gossiping. In fact, this work and subsequent ones such as [12][9][17][5] mainly focus on the successive translations of queries, on evaluation of the quality of translation as it may induce some losses and on progressive learning of other peers' schemas. For example, in [17], peers update their one-hop neighborhood according to the accuracy of the answers they receive. Some works use an explicit representation of overlays such as [23][17]. In [23], an overlay is represented by a set of concepts (from a reference ontology). The explicit description of the overlays has to be maintained for new peers to be able to join overlays. A more flexible solution is proposed in [17] which automatically computes schema synopses from semantic clusters. On the contrary, we use gossiping, with no explicit representation of overlays.

As a conclusion, our work benefits from previous contributions to provide a new original approach to build a totally decentralized, two layered solution for semantically heterogeneous P2P systems for distributed information retrieval where peers annotate their documents with an ontology. A main characteristic is the total independence of the two

software layers. Indeed, the way structural heterogeneity is decreased is totally disconnected from the way documents are retrieved. Together with the definition of gossip, this is a major difference with the works previously described. Because they mix query evaluation and learning about peers' schemas, it is possible to study their global performances but no one can highlight the individual performances of the learning-of-mappings solution nor those of the clustering algorithm in reducing semantic heterogeneity.

VII. Conclusion

In the context of P2P information retrieval systems where peers annotate their resources w.r.t. ontologies, we proposed to distinguish semantic heterogeneity and interoperability. The former is viewed as a characteristic of the system, that depends on the ontologies used, on the peers' semantic knowledge and on their relationships. It represents the difficulty of the problem. Interoperability is more application dependent and is linked to the retrieval performances, that can be measured through usual metrics such as precision and recall.

To illustrate this approach, we defined two independent and complementary protocols: (i) the GoOD-TA protocol that reduces the semantic heterogeneity related to the topology by putting closer peers that can understand each others and (ii) the DiQuESH protocol, a distributed top-k algorithm that ensures some interoperability. First we showed the nice behavior of the GoOD-TA protocol in reducing semantic heterogeneity, handling dynamicity and the evolution of peers' descriptors. Then we showed that precision and recall values obtained by the DiQuESH protocol alone are improved if we run both algorithms together. This enables showing the indirect contribution of GoOD-TA to interoperability.

Future work encompasses more experiments with other data sets and definition of additional algorithms to manage semantic heterogeneity/interoperability. We currently assume no peers' malicious behaviors, such as cheating about one's profile or one's own documents relevance score. We plan to include existing solutions that enable to avoid such behaviors.

References

[1] Karl Aberer, Philippe Cudré-Mauroux, and Manfred Hauswirth. The chatty web: emergent semantics through gossiping. In *WWW*, pages 197–206, 2003.

[2] Reza Akbarinia, Esther Pacitti, and Patrick Valduriez. Reducing network traffic in unstructured P2P systems using top-k queries. *Distributed and Parallel Databases*, 19(2-3):67–86, 2006.

[3] Grigoris Antoniou and Frank van Harmelen. Web ontology language: OWL. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 91–110. Springer Berlin Heidelberg, 2009. Second Edition.

[4] Marin Bertier, Davide Frey, Rachid Guerraoui, Anne-Marie Kermarrec, and Vincent Leroy. The gossple anonymous social network. In *11th International Middleware Conference*, pages 191–211, 2010.

[5] Angela Bonifati, Elaine Qing Chang, Terence Ho, Laks V. S. Lakshmanan, Rachel Pottinger, and Yongik Chung. Schema mapping and query translation in heterogeneous P2P XML databases. *VLDB Journal*, 19(2):231–256, 2010.

[6] Thomas Cerqueus, Sylvie Cazalens, and Philippe Lamarre. Gossiping correspondences to reduce semantic heterogeneity of unstructured P2P systems. In *4th International Conference on Data Management in Grid and P2P Systems*, pages 37–48, 2011.

[7] Thomas Cerqueus, Sylvie Cazalens, and Philippe Lamarre. Semantic heterogeneity measures of unstructured P2P systems. In *10th IEEE/WIC/ACM International Conference on Web intelligence*, 2011.

[8] Carmela Comito, Simon Patarin, and Domenico Talia. A semantic overlay network for P2P schema-based data integration. In *11th IEEE Symposium on Computers and Communications*, pages 88–94, 2006.

[9] Philippe Cudré-Mauroux, Suchit Agarwal, and Karl Aberer. GridVine: An infrastructure for peer information management. *IEEE Internet Computing*, 11(5):36–44, 2007.

[10] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, 2007.

[11] Natalya Fridman Noy, Nigam H. Shah, Patricia L. Whetzel, Benjamin Dai, Michael Dorf, Nicholas Griffith, Clément Jonquet, Daniel L. Rubin, Margaret-Anne D. Storey, Christopher G. Chute, and Mark A. Musen. Bioportal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research*, 37(Web-Server-Issue):170–173, 2009. `http://bioportal.bioontology.org`.

[12] Alon Halevy, Zachary Ives, Peter Mork, and Igor Tatarinov. Piazza: data management infrastructure for semantic web applications. In *12th International World Wide Web Conference*, pages 556–567, 2003.

[13] Michael Hartung, James F. Terwilliger, and Erhard Rahm. Recent advances in schema and ontology evolution. In *Schema Matching and Mapping*, pages 149–190. Springer, 2011.

[14] Márk Jelasity and Ozalp Babaoglu. T-man: Gossip-based overlay topology management. In *In 3rd International Workshop on Engineering Self-Organising Applications (ESOA'05)*, pages 1–15. Springer-Verlag, 2005.

[15] Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In *5th ACM/IFIP/USENIX International Middleware Conference*, pages 79–98, 2004.

[16] Márk Jelasity, Alberto Montresor, and Özalp Babaoglu. T-Man: Gossip-based fast overlay topology construction. *Computer Networks*, 53(13):2321–2339, 2009.

[17] Verena Kantere, Dimitrios Tsoumakos, Timos K. Sellis, and Nick Roussopoulos. GrouPeer: Dynamic clustering of P2P databases. *Information Systems*, 34(1):62–86, 2009.

[18] Anne-Marie Kermarrec and Maarten van Steen. Gossiping in distributed systems. *Operating Systems Review*, 41(5):2–7, 2007.

[19] Alexander Löser, Felix Naumann, Wolf Siberski, Wolfgang Nejdl, and Uwe Thaden. Semantic overlay clusters within super-peer networks. In *1st International Workshop, Databases, Information Systems, and Peer-to-Peer Computing*, pages 33–47, 2003.

[20] Alberto Montresor and Márk Jelasity. Peersim: A scalable P2P simulator. In *9th IEEE International Conference on Peer-to-Peer Computing*, pages 99–100, 2009.

[21] Andronikos Nedos, Kulpreet Singh, Raymond Cunningham, and Siobhán Clarke. A gossip protocol to support service discovery with heterogeneous ontologies in manets. In *3rd IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, page 53, 2007.

[22] Wolfgang Nejdl, Martin Wolpers, Wolf Siberski, Christoph Schmitz, Mario T. Schlosser, Ingo Brunkhorst, and Alexander Löser. Super-peer-based routing strategies for RDF-based peer-to-peer networks. *Journal of Web Semantics*, 1(2):177–186, 2004.

[23] Wilma Penzo, Stefano Lodi, Federica Mandreoli, Riccardo Martoglia, and Simona Sassatelli. Semantic peer, here are the neighbors you want! In *EDBT*, pages 26–37, 2008.

[24] Gerard Salton, Andrew Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18:613–620, 1975.

[25] Stefan Saroiu, Krishna P. Gummadi, and Steven D. Gribble. Measuring and analyzing the characteristics of Napster and Gnutella hosts. *Multimedia Systems*, 9(2):170–184, 2003.

[26] Damires Souza, Carlos Eduardo S. Pires, Zoubida Kedad, Patricia Azevedo Tedesco, and Ana Carolina Salgado. A semantic-based approach for data management in a P2P system. *Transactions on Large-Scale Data- and Knowledge-Centered Systems*, 3:56–86, 2011.

[27] Spyros Voulgaris, Anne-Marie Kermarrec, Laurent Massoulié, and Maarten van Steen. Exploiting semantic proximity in peer-to-peer content searching. In *10th IEEE International Workshop on Future Trends in Distributed Computing Systems*, pages 238–243, 2004.