LABORATOIRE D'INFORMATIQUE DE NANTES ATLANTIQUE

Mise en place de la méthode ExSI^{2}D dans un système P2P hétérogène

Thomas CERQUEUS

encadré par Sylvie CAZALENS et Patrick VALDURIEZ

Équipe ATLAS-GDD

RAPPORT DE STAGE DE MASTER 2 ALMA

Septembre 2009

LINA, Faculté des Sciences & Techniques de Nantes
2, rue de la Houssinière
B.P. 92208 — F-44322 NANTES CEDEX 3

UNIVERSITÉ DE NANTES

Thomas CERQUEUS (encadré par Sylvie CAZALENS et Patrick VALDURIEZ)

Mise en place de la méthode ExSI^{2}D dans un système P2P hétérogène

Mise en place de la méthode ExSI^{2}D dans un système P2P hétérogène

Thomas CERQUEUS (encadré par Sylvie CAZALENS et Patrick VALDURIEZ)

thomas.cerqueus@etu.univ-nantes.fr

Remerciements

Je tiens tout d'abord à remercier Patrick VALDURIEZ et Sylvie CAZALENS de m'avoir proposé ce sujet de stage, et de m'avoir encadré avec dynamisme durant les six mois qui viennent de s'écouler. Merci aussi à Philippe LAMARRE et Anthony VENTRESQUE pour le temps qu'ils m'ont accordé et pour les conseils qu'ils m'ont donné. Merci aussi à William DEDZOÉ avec qui les échanges sont enrichissants. J'ai vraiment pris du plaisir à venir travailler avec vous.

Je voudrais aussi remercier l'ensemble de l'équipe Atlas-GDD pour l'accueil qui m'a été fait. Merci également à ceux qui m'ont encouragé à choisir la recherche (Gerson, Eduardo, ...).

Enfin je voudrais dire merci à ceux qui m'ont soutenu durant tout mon parcours universitaire et sans qui je n'aurai pas pu arriver ici : Papa, Maman, Damien, Audrey et Sylvie.

Table des matières

Chapitre 1

Introduction

1.1 Présentation du contexte

Depuis quelques décénnies, la quantité de données numériques augmente de façon importante et le phénomène ne fait que s'accentuer avec le développement d'Internet. Le volume d'information est si important, qu'il devient de plus en plus difficile de trouver de l'information pertinente : les systèmes de recherche d'information (RI) sont indispensables et les difficultés auxquelles ils sont confrontés sont d'autant plus délicates.

Les systèmes de recherche d'information les plus connus (Google, Yahoo) sont des systèmes centralisés : les données sont stockées sur des serveurs centraux. Ce type d'architecture (client/serveur) présente des limites : pas de tolérance aux pannes et difficulté à passer à l'échelle. Les systèmes pair-à-pair (P2P) ont été proposés pour répondre à ces problématiques, mais la distribution des données dans de tels systèmes fait apparaître de nouveaux défis pour les systèmes de RI : recherche des données dans le réseau, placement des pairs, hétérogénéité entre les pairs, etc.

Le contexte dans lequel nous allons travailler se trouve à la croisée de ces deux domaines : la recherche d'information et les systèmes P2P.

1.2 Problématiques abordées et objectifs

Ce stage a pour cadre général la définition de systèmes P2P pour le partage d'informations. Chaque participant gère ses informations de manière autonome et répond aux requêtes qui lui parviennent ; il peut effectuer lui-même des recherches dans le système.

Nous nous plaçons dans le cas où les informations partagées sont des documents. Ceux-ci sont représentés par des vecteurs sémantiques : pour un document donné, chaque concept d'une ontologie est pondéré selon sa représentativité dans le document. La pondération des concepts des vecteurs sémantiques est assurée par le processus d'indexation. Comme les participants peuvent être nombreux, il est difficile de concevoir qu'ils puissent convenir de l'utilisation d'une seule et même ontologie (même pour un domaine particulier). Ainsi les ontologies des participants peuvent différer : il y a donc hétérogénéité sémantique.

Dans ce contexte, A. Ventresque *et al.* [34] proposent la méthode ExSI^{2}D. Cette méthode utilise trois modules qui assurent les fonctionnalités suivantes : expansion structurante de requêtes, interprétation et calcul d'image de documents. La méthode ExSI^{2}D a été définie dans un contexte où seulement deux participants interviennent : un initiateur de requêtes et un fournisseur de documents.

Le premier objectif est la mise en place de cette méthode dans un système P2P et de sa simulation avec PeerSim. Pour atteindre cet objectif, il faudra que nous fassions des choix concernant entre autres : le type de réseau P2P que nous utiliserons, les algorithmes de routage des requêtes, la distribution des données d'un corpus de test dans le système P2P de manière réaliste.

Le second objectif est de proposer des mesures permettant de quantifier le degré d'interopérabilité sémantique entre deux pairs utilisant des ontologies. Ces mesures devront permettre d'améliorer le processus de recherche d'information dans les systèmes P2P hétérogènes sémantiquement. Elles pourront par exemple être utilisées lors du placement des pairs dans le réseau (choix des voisins, clusterisation) ou lors du routage des requêtes.

1.3 Plan du rapport

Dans un premier temps (chapitre 2), nous allons présenter un état de l'art sur les systèmes P2P, sur la recherche d'information et sur l'interopérabilité sémantique. Dans cette partie, nous allons tenter de présenter les notions et les problématiques liées aux sujets que nous abordons.

Ensuite, dans le chapitre 3, nous présenterons les mesures d'interopérabilité sémantique que nous proposons. Ces mesures utilisent des alignements d'ontologies pour quantifier le degré d'interopérabilité entre ontologies.

Le chapitre 4 présente le travail réalisé pour mettre en place la méthode ExSI^{2}D dans un cadre distribué. Dans cette partie, nous abordons aussi bien le travail de modélisation et de développement que le travail d'expérimentation.

Nous terminerons par faire un bilan sur le travail réalisé et sur les perspectives ouvertes par celui-ci (chapitre 5).

Chapitre 2

État de l'art

Étant donné que le sujet du stage se positionne à la croisée de deux domaines principaux (la recherche d'information et les systèmes P2P), nous devons avoir de bonnes connaissances sur ces deux domaines : c'est ce que nous abordons dans ce chapitre.

Pour commencer, la section 2.1 présente, de manière assez générale, les systèmes P2P. Ensuite, la section 2.2 présente la recherche d'information (les mesures de qualités usuelles, les modèles existants, etc.). Dans la section 2.3 nous abordons la notion d'interopérabilité sémantique (hétérogénéité sémantique, alignement d'ontologies, méthode ExSI^{2}D).

2.1 Systèmes P2P

2.1.1 Types de réseaux

Les systèmes pair-à-pair (P2P) sont des systèmes distribués conçus pour le partage de ressources (données, capacité de calcul) en s'affranchissant du modèle client/serveur. Leur objectif est d'obtenir une meilleur tolérance aux pannes et un meilleur passage à l'échelle.

Les principales caractéristiques des systèmes P2P sont les suivantes :
- La gestion du systèm P2P est décentralisée : il n'y a pas de serveur central et les pairs sont distribués sur un large réseau.
- Les pairs sont volatiles : un pair peut rejoindre ou quitter le système à tout moment.
- Les pairs sont autonomes : ce sont les pairs qui décident de partager (ou non) leurs ressources et de rejoindre/quitter le système.

Trois classes de réseaux P2P sont généralement considérées : non-structuré, stucturé et super-pair.

Non-structurés

Les systèmes non-structurés sont créés de manière aléatoire : lorsqu'un pair arrive dans le système, il contacte aléatoirement des pairs. Ces pairs deviennent ses voisins. Le placement des données dans le réseau n'est pas déterminé par la topologie. Aucun pair ne jouant de rôle central (ou privilégié), la tolérance aux pannes est très importante.

Le routage de message est généralement effectué par inondation (*flooding*) du réseau : le pair initiateur du message envoie le message à tous ses voisins. Lorsque ces derniers reçoivent le message, ils le traitent et le transmettent à leurs propres voisins. Le message est ainsi envoyé à tous les pairs du système. Ce processus ne permet pas de passer à l'échelle facilement car le nombre de messages échangés est très important. Il peut y avoir saturation du réseau. Gnutella [23] et FreeHaven [3] sont des réseaux P2P non-structurés.

Structurés

Les réseaux structurés ont été proposés pour faire face au problème de passage à l'échelle des réseaux non-structurés. Ils contrôlent directement la topologie et le placement des données pour arriver à diminuer fortement le nombre de messages échangés. En effet les données sont placées à des positions précises : il est donc plus facile de les retrouver.

Les réseaux structurés prennent très souvent la forme de tables de hachage distribuées (*distributed hash table* : DHT).Ces tables permettent d'associer une clé à tout objet à indexer dans le système P2P. Les pairs sont responsables de données qui leurs sont attribuées suivant leur position dans le réseau. Pour trouver un objet, il suffit donc de connaître la clé à laquelle il correspond, ce qui donne la position où il se trouve, et donc le pair qui en est responsable.

Le routage des requêtes est très efficace : trouver le pair responsable d'un object se fait en $O(\log n)$ décisions de routage, n étant le nombre de pairs dans le réseau. L'autonomie est cependant limitée, autant dans les données mises à disposition par les pairs que par leur position dans le réseau. Chord [1], Pastry [9] et P-Grid [8] sont des réseaux P2P structurés.

Super-pairs

Ils sont appelés hybrides en référence au fait qu'ils sont un compromis entre les réseaux P2P "purs" (où tous les pairs sont égaux et fournissent les mêmes fonctionnalités) et les réseaux clients/serveurs. Dans ces réseaux, certains pairs sont choisis comme *super-pairs* et ont le rôle de serveurs pour d'autres pairs. Les super-pairs eux-mêmes sont aussi en relation de voisinage entre eux, et peuvent utiliser n'importe quel système P2P pour s'organiser. Les super-pairs peuvent être élus par rapport à leurs capacités (bande passante, capacité de calcul) et remplacés dynamiquement.

Le temps pris pour trouver une donnée parmi les super-pairs (qui indexent les pairs dont ils sont responsables) est très court, comparé à l'inondation. Les réseaux de ce type sont cependant très dépendants des super-pairs qui les composent : ces derniers regroupant tous les problèmes des serveurs dans les architectures client/serveurs. Edutella [2] et JXTA [10] sont des réseaux super-pairs.

2.1.2 Routage de requêtes dans des systèmes non-structurés

Le routage de messages dans les systèmes P2P est une tâche centrale et difficile. Les approches classiques sont gourmandes en nombre de messages envoyés sur le réseau. Les sections suivantes décrivent différentes approches qui essaient de minimiser cette quantité.

Quelques algorithmes de routage

BFS Dans sa version de base, la recherche par parcours en profondeur (*breath-first search*, BFS) inonde le réseau P2P à une distance TTL (*Time-To-Live*) de l'initiateur de la requête. Chaque pair recevant la requête décroît la valeur du TTL, exécute la requête et la transmet à tous ses voisins (si $TTL \geqslant 1$). Cela permet donc d'atteindre tous les pairs situés à une distance TTL de l'initiateur. Des améliorations de cette méthode ont été proposées pour limiter le nombre de messages envoyés (par exemple [25]).

Random walks L'algorithme de marche au hasard (*random walks*) génère k "marcheurs" qui partent de différents voisins du pair initiateur de la recherche et parcourent le réseau. À chaque fois qu'un marcheur atteint un pair, ce dernier exécute la requête et interroge le pair initiateur pour savoir si la condition de terminaison est remplie. Si elle n'est pas remplie, le marcheur se dirige vers l'un des voisins du pair courant. Il est aussi possible de n'interroger que régulièrement le pair d'origine. L'efficience de cet algorithme est assez bonne au sens où le nombre total de messages ($k \times TTL$ dans le pire des cas) n'est pas dépendant de la topologie. Par contre les résultats dépendent de la topologie et des choix de routage (aléatoires).

Algorithmes Top-K

Les techniques présentées précédemment concernent principalement les requêtes qui cherchent un sous-ensemble de solutions, dont la condition est unique. Dans certains cas (par exemple pour la surveillance de système et réseau, la recherche d'information ou les bases de données multimédia), l'initiateur de la requête n'est pas intéressé par tous résultats : les meilleurs suffisent. Les requêtes *k-meilleurs* (ou requêtes *top-k*) permettent d'exprimer ce besoin en indiquant un nombre k de résultats dont la pertinence est la plus forte. La valeur de k est fixée par le pair initiateur de la requête. Les algorithmes Top-K permettent de traiter (ie. transmettre dans le système P2P) ce type de requête.

La solution naïve consiste à envoyer la requête à tous les pairs accessibles, à l'exécuter et retourner les réponses correctes à l'initiateur qui fait le classement. Il est évident que cette solution n'est pas acceptable en coût de communication et temps de traitement.

Akbarinia *et al.* [11] proposent une solution totalement distribuée (fully distributed, FD) pour exécuter des requêtes top-k dans des réseaux structurés. L'algorithme débute sur le pair P_i : le pair initiateur de la requête Q. Celui-ci commence par :

1. Initialiser la valeur du TTL.
2. Donner un identifiant unique QID à la requête Q. QID est déterminé à partir de l'identifiant du pair P_i et d'un compteur global de requêtes. Les pairs vont pouvoir utiliser cet identifiant pour distinguer les nouvelles requêtes et celles déjà reçues.

Le pair P_i va maintenant devoir exécuter les quatre phases suivantes : transmission de la requête, exécution locale de la requête, fusion et renvoie des résultats et récupération des données.

Transmission de la requête Lorsqu'un pair P reçoit un message incluant la requête Q, les actions suivantes sont effectuées :

1. si la requête a déjà été reçue (la vérification se fait grâce à l'identifiant QID), le message est ignoré ; sinon l'adresse de l'expéditeur est conservée : l'expéditeur devient le père de P.

2. la valeur du TTL est diminuée de 1 : si elle reste strictement positive, un nouveau message est créé et envoyé à tous les voisins de P (sauf à ses parents). Le message créé contient la requête Q, son identifiant QID, la nouvelle valeur du TTL et l'adresse de l'initiateur de la requête P_i.

Exécution locale de la requête Après avoir procédé à la transmission de la requête, chaque pair exécute Q localement. L'exécution de la requête se traduit par le parcours de toutes les données du pair et par l'attribution d'une valeur de pertinence à chaque donnée. La pertinence d'une donnée d par rapport à une requête Q est mesurée grâce à une fonction de score (*scoring function*) notée $Sc(d, Q)$.

Une fois que les k meilleurs données locales sont sélectionnées, le pair doit attendre les réponses de ses voisins avant de commencer la phase suivante. Pour éviter d'attendre indéfiniment les réponses d'un pair ayant quitté le système, il faut fixer une limite de temps d'attente (notons le WT). Cette limite est calculée en fonction de la valeur du TTL, des paramètres du système et des paramètres du pair.

Fusion et envoi des résultats Une fois le temps d'attente WT écoulé, le pair procède à la fusion entre ses résultats (les k meilleurs) et ceux de ses voisins. Il envoie le résultats à son père (celui qui lui avait transmis la requête) sous forme d'une liste de scores (*score-list*).

Pour minimiser le traffic sur le réseau, la liste de scores ne contient pas les données elles-mêmes. Si la donnée d, contenue dans le pair p, fait partie des k meilleurs réponses alors le couple (p, $Sc(d, Q)$) appartient à la liste de scores.

Récupération des données L'initiateur de la requête commence par produire la liste finale des scores : L_f. Chaque élément de la liste est un couple (p, s), où p est l'adresse d'un pair et s une valeur de pertinence. Pour récupérer les k meilleurs données, l'initiateur de la requête n'a plus qu'à :

1. déterminer le nombre de fois où le pair p appartient à la liste des scores. Notons ce nombre m.
2. demander au pair p de lui retourner ses m meilleurs données.

Le pair obtient ainsi les k données les plus pertinentes, dans un rayon R (où R est la valeur du TTL fourni par le pair initiateur).

Cet algorithme est intéressant car il permet de gérer l'envoie de requêtes et la réception des résultats dans les systèmes P2P. Il sera nécessaire pour la mise en place de la méthode ExSI^{2}D dans un système P2P.

2.1.3 Simulation d'un système P2P

Motivations

Les spécificités des systèmes P2P (la distribution, la volatilité et l'autonomie des pairs) font qu'il est difficile de prévoir leur comportement en situation réelle.

Il est certe possible de mettre en place un véritable réseau P2P, en utilisant Grid5000 [4] par exemple, mais cela est parfois compliqué. Dans le cas de Grid5000, un certain nombre de règles de “savoir-vivre” sont en place, et cela ne facilite pas l’utilisation du réseau de cluster. Ces difficultés ont déjà été rencontrées lors du projet de fin d’étude [33] auquel j’ai participé. Une des solutions est de simuler un système P2P : l’exécution est alors centralisée mais les caractéristiques d’un système distribué sont conservées.

Pour choisir un outil de simulation, il faut évaluer les besoins. En effet, chaque outil possède ses propres caractéristiques : sa capacité à passer à l’échelle, sa dynamicité, ses performances (en temps et en ressources).

Outils de simulations existants

Kotilainen *et al.* [26] propose une comparaison entre différents outils de simulation : NS-2, QueryCyle, 3LS, PeerSim, GPS (*General P2P Simulator*), etc. Le tableau ci-dessous présente certaines de leurs caractéristiques.

	# pairs gérés	Dynamique	Langage de prog.
NS-2	< 1000	Non	C++
QueryCycle	-	Oui	Java
3LS	< 1000	Oui	Java
PeerSim	10^6	Oui	Java
NeuroGrid	300000	Oui	Java
GPS	-	Oui	Java

TAB.1 Caractéristiques de quelques outils de simulation.

PeerSim [21] possède des propriétés intéressantes : il supporte un grand nombre de pairs et il gère l’entrée et la sortie de pairs en cours d’exécution (cf. tableau 1). De plus, il offre de bonnes fonctionnalités : par exemple il permet de paramétrer la connectivité des pairs (ie. le nombre de voisins) et la répartition des données sur le réseau. C’est pour ces raisons que l’équipe Atlas-GDD du LINA[1] a choisi de l’utiliser.

Il faut tout de même noter que la version actuelle (celle disponible sur le site du projet), ne gère pas la charge réseau. Cela peut devenir gênant car il est évident que les performances d’un système distribué dépendent de la charge des pairs. Deux modules complémentaires ont été développés par l’équipe Atlas-GDD pour permettre la gestion de la charge. Le premier module gère la charge des pairs en ajoutant des délais de travail lors de l’exécution de certaines méthodes (par exemple : accèder à une base de données ou à des documents stockés sur disque). Le second module gère la charge du réseau en considérant que le système ne peut supporter qu’un certain nombre de messages à un instant donné. La prise en compte de la charge est un peu grossière mais elle permet déjà de voir apparaître des phénomènes de surcharge réseau.

[1] Laboratoire d’Informatique de Nantes Atlantique

La mise en place de la méthode ExSI^{2}D nécessite de disposer d'un système P2P et d'un système de recherche d'information. La mise en place du réseau de pairs sera effectué grâce à l'outils de simulation PeerSim. Il faut désormais que nous traitions la partie relative à la recherche d'information.

2.2 Recherche d'information

2.2.1 Généralités

Une définition de la recherche d'information

Le but d'un système de recherche d'information (RI) est de répondre aux besoins en information d'un utilisateur. Ce dernier peut exprimer ses besoins sous différentes formes ; aussi bien à l'aide d'une liste de mots-clés, que sous forme d'une requête en langage naturel. Le système de RI doit fournir des documents répondant aux besoins exprimés par l'utilisateur.

Selon Boughanem [15], en recherche d'information, les notions clés sont les suivantes : les documents (ie. l'information), le besoin en information (exprimé sous forme d'une requête) et la pertinence (proximité de la requête avec un document).

Plusieurs difficultés font que la recherche d'information est un exercice complexe. Tout d'abord il faut être capable de comprendre le besoin exprimé par l'utilisateur. Et ensuite il faut trouver les documents pertinents tout en minimisant le nombre de documents non pertinents.

La recherche d'information basée sur la recherche de mots-clés dans les documents fonctionne bien mais a des limites. La recherche d'information sémantique tente de dépasser ces limites en utilisant la sémantique (en considérant la notion de *concept* plutôt que celle de *mot-clé*). Elle permet entre autre de désambiguiser les requêtes et les documents afin de retrouver davantage d'information pertinente. La sémantique est apportée par l'utilisation d'ontologies. Aussenac-Gilles [15] définit une ontologie comme la modélisation de connaissances d'un domaine (plus ou moins grand) sous forme d'un réseau de concepts. L'organisation des concepts est choisie de manière à favoriser leur classification : la structure d'une ontologie comporte donc systématiquement une hiérarchie de spécialisation des concepts (lien de subsomption : *is-a*). D'autres relation entre concepts sont aussi fréquemment utilisées : composition, disjonction, etc.

Mesures de qualité

Pour évaluer un système de RI, il est nécessaire de pouvoir mesurer la qualité des réponses fournies par celui-ci par rapport aux besoins exprimés par l'utilisateur dans sa requête. Nous avons donc besoin de mesures de pertinence.

Il y a de nombreuses façons de mesurer la qualité d'un système de recherche d'information. Les mesures les plus courantes sont le *rappel* et la *précision*. Avant de présenter ces mesures, nous devons introduire quelques notations.

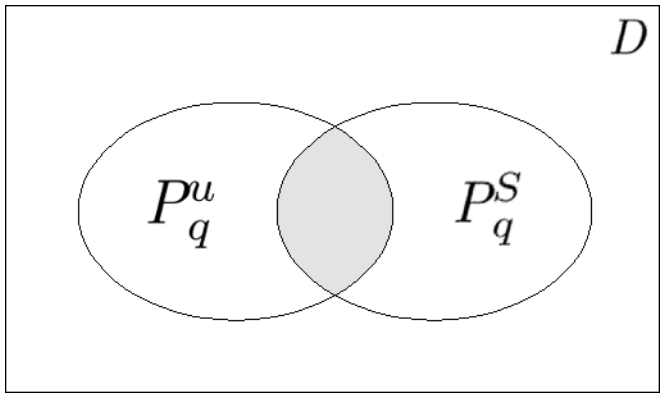

FIG. 2.1 – L'ensemble des documents sélectionnés par le système S (P_q^S) et l'ensemble des documents pertinents pour l'utilisateur u (P_q^u) parmi l'ensemble des documents proposés D.

Soit *D* un ensemble de documents, et q une requête posée sur cet ensemble. Pour cette requête, un sous-ensemble des documents peut être dit *pertinent* pour un utilisateur u, et un autre *non pertinent*. Notons ces deux ensembles $P_q^u \subseteq D$ et $\overline{P}_q^u = D \setminus P_q^u$.

Tout système *S* sélectionne des documents appartenant à *D* comme réponses adéquates pour la requête *q*. Nous pouvons noté l'ensemble de ces documents P_q^S.

Il existe un certain nombre de mesures pour calculer la qualité d'un système de RI (*F-mesure*, taux de chute, etc.), mais les deux mesures les plus connues sont certainement le rappel et la précision. Ce sont d'ailleurs celles-ci qui ont été utilisées pour évaluer la méthode ExSI^{2}D.

Le rappel Il représente le rapport entre le nombre de documents pertinents retournés et le nombre de documents pertinents. Une valeur à 1 signifie que tous les documents pertinents ont été trouvés. Il se calcule d'après la formule suivante :

$$R_q = \frac{\|P_q^u \cap P_q^S\|}{\|P_q^u\|}$$

La précision Elle représente le rapport entre le nombre de documents pertinents retournés et le nombre de documents retournés. Une valeur à 1 signifie que tous les documents retournés sont pertinents (ie. il n'y a pas de bruit). La précision se calcule grâce à la formule suivante :

$$P_q = \frac{\|P_q^u \cap P_q^S\|}{\|P_q^S\|}$$

L'idéal est lorsque les deux mesures sont égales à 1 ; dans ce cas on a $P_q^S = P_q^u$. En pratique, aucun système ne satisfait cette condition. Les tests de *Cranfield* [13] ont montré que lorsqu'un système obtient un bon taux de rappel, le taux de précision est faible et inversement. La figure 2.2, réalisée d'après la figure 8.2 de [13], montre :

– un système qui a un bon taux de rappel mais une faible précision (point A),

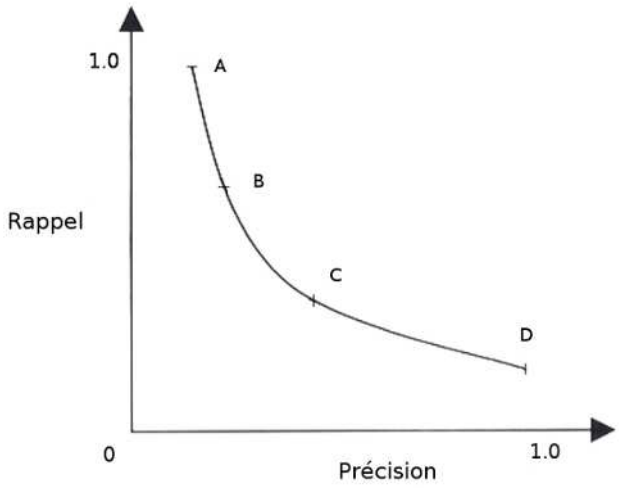

FIG. 2.2 – Courbe caractéristique des taux de précision et de rappel. Les systèmes ayant un bon taux de rappel, ont généralement une mauvaise précision (système A) et inversement (système D). Certains systèmes se trouvent entre les deux (systèmes B et C).

– un système qui a une bonne précision mais un faible taux de rappel (point D),
– des systèmes qui cherchent un compromis entre rappel et précis (point B et C).

En général la précision est préférée au rappel ([17]).

Corpus de test

À présent, nous devons définir le contexte d'évaluation, c'est-à-dire les éléments qui vont servir à tester le processus de sélection des documents pour chaque système. En recherche d'information, il s'agit d'un *corpus de test*. Un corpus est composé de :

– un ensemble de documents : c'est sur cet ensemble que vont être posées les requêtes.
– un ensemble de requêtes : elles simulent l'activité d'un utilisateur.
– des jugements de pertinence : pour chaque requête, ils déterminent quels sont les documents pertinents (parfois le degré de pertinence est également indiqué).

Gregor [19] propose une description d'un certain nombre de corpus libres : 20newsgroups, Reuters-21578, RCV-1v2, TREC-AP, NIPS, PNAS, CiteSeer, etc. Le tableau ci-dessous présente quelques-uns de ces corpus avec leurs caractéristiques :

Nom du corpus	# documents	# requêtes
Cranfield	1 400	225
CISI	1 460	112
NPL	11 429	93
20newsgroups	20 000	-
TREC-AP	209 783	-
TREC-W3C	331 037	99

TAB.2 Quelques corpus de test.

Le corpus choisi pour tester ExSI^{2}D est Cranfield pour plusieurs raisons. Premièrement, comme notre travail porte sur le traitement des requêtes, nous devons avoir un échantillon suffisamment important. Cranfield est l'un des corpus qui propose le plus de requêtes : il répond bien à ce que nous cherchons. Deuxièmement, nous voulons effectuer des expérimentations avec les données du corpus. Le temps d'exécution doit être relativement court et les ressources utilisées (mémoire, espace disque, etc.) doivent être raisonnables. Le fait que Cranfield contienne seulement 1400 documents (ce qui peut paraître peu par rapport à d'autres corpus : TREC, etc.) semble donc être une bonne chose.

2.2.2 Méthodes de recherche d'information

Indexation

L'indexation est le processus qui consiste à extraire les informations importantes d'un document ou d'une requête. L'objectif est d'obtenir, pour chaque document et chaque requête, une description refletant leurs contenus. L'indexation peut être réalisée manuellement ou bien de manière automatique.

Dans le cas de l'indexation manuelle, il faut que celle-ci soit réalisée par des spécialistes du domaine abordé dans les documents. En effet, pour obtenir une indexation de bonne qualité, il faut que les termes de l'index soient choisis avec précision. Le coût d'une telle indexation est important, d'autant plus lorsque la collection de documents à indexer est grande.

L'indexation automatique est bien moins coûteuse, et fournit des résultats similaires à ceux obtenus manuellement. Baeza-Yates et Ribeiro-Neto [12] assimilent l'indexation à un pré-traitement des documents. Les phases clés du pré-traitement sont :

L'élimination des mots vides de sens Les mots vides de sens (*stop words*) sont tellement communs qu'il ne sert à rien de les utiliser : « le », « la », « du », « ça », etc. Ces mots vides sont spécifiques à une langue et/ou une collection de documents.

La lemmatisation Cette étape, aussi appelée racinisation, consiste à ramener les termes à une racine commune. Par exemple, les termes « précieuse », « précieuses » ou « préciosité », sont transformés en une racine simple : « précieux ».

La sélection des termes de l'index En général les termes choisis sont des noms car ils sont plus significatifs que les verbes et les adjectifs.

La pondération des termes Elle se base souvent sur les facteurs tf (la fréquence d'apparition d'un terme dans un document) et idf (la fréquence d'apparition de ce même terme dans toute la collection considérée) qui permettent de considérer les pondérations locales et globales d'un terme. La mesure $tf * idf$ permet d'approximer la représentativité d'un terme dans un document, surtout dans les corpus de documents de tailles homogènes.

L'indexation peut être lexicale (l'index est constitué de mots-clés) ou sémantique (l'index est constitué de concepts).

Modèles de recherche d'information

Un modèle de recherche d'information sert à définir la représentation des documents et des requêtes, ainsi que la manière de les comparer (ie. définir la fonction de comparaison). Dans la suite de cette section, sont présentés quatre modèles de RI : le modèle booléen, le modèle probabiliste, le modèle vectoriel et le modèle vectoriel sémantique.

Le modèle booléen Le *modèle booléen* représente les documents comme des ensembles de termes et les requêtes comme des expressions booléennes (composées d'opérateurs booléens) sur ces termes. Il est aussi possible d'utiliser des concepts plutôt que des termes dans ce modèle. C'est le modèle le plus utilisé dans les moteurs de recherche sur le Web. Les opérateurs booléens les plus courants sont la négation, la disjonction et la conjonction. Parfois sont aussi utilisés des opérateurs un peu plus riches, comme des opérateurs d'ordre : *before*, *after*. La fonction de pertinence renvoie, pour un document et une requête donnés, la valeur *Vrai* ou *Faux* selon que le document est jugé pertinent ou non. L'évaluation est faite selon les fonctions d'interprétation usuelles des opérateurs booléens.

Parmi les avantages de ce modèle, nous pouvons citer sa grande facilité de mise en œuvre, l'efficacité du calcul, ainsi que l'expressivité et la clarté des requêtes, basées sur une logique booléenne. C'est pourquoi les moteurs de recherche l'ont choisi de manière évidente : il permet une recherche rapide, et il a une syntaxe apparemment simple.

Ce modèle n'a pas que des avantages. En effet, la syntaxe et la sémantique des requêtes est finalement assez compliquée et pas toujours très intuitive. La plupart des requêtes sont interprétées comme booléennes par les interfaces de recherche des moteurs de recherche : il s'agit la plupart du temps d'un ensemble de mots-clés traduit en une requête booléenne (conjonction des termes). D'autre part, le modèle est par défaut sur le mode "tout ou rien". La valeur 1 est donnée aux documents qui satisfont la formule booléenne de la requête, et 0 aux autres. Ce qui ne permet évidemment pas un classement des documents, mais renvoie juste la liste des documents adéquats. Il n'y a pas non plus dans le modèle standard de pondération des termes des documents ou des requêtes. Bien évidemment, certains des problèmes décrits ici sont adressés, d'une façon ou d'une autre, par les systèmes de RI.

Le modèle probabiliste Le modèle probabiliste a été développé dans les années Soixante-dix, mais a connu des développements récents, car les approches basées sur ce modèle ont obtenu de très bons résultats dans TREC, par exemple le système OKAPI [32]. Dans le modèle probabiliste, la fonction de pertinence utilise les probabilités. Soit $P(R|d_i)$ la probabilité que le document d_i soit pertinent pour la requête q, et $P(\overline{R}|d_i)$ la probabilité que ce document ne soit pas pertinent pour cette requête. Alors la pertinence entre le document d et la requête q est :

$$sim(d,q) = \frac{P(R|d_i)}{P(\overline{R}|d_i)}$$

Ce modèle est assez lourd à mettre en œuvre, car il nécessite d'avoir une estimation des probabilités initiales. Néanmoins le cadre est très solide mathématiquement, et il permet un grand nombre de techniques formellement éprouvées et qui pratiquement, sont prometteuses.

Le modèle vectoriel Le modèle vectoriel (*vector space model*) est sans doute un des modèles les plus connus en RI. Il consiste à représenter documents et requêtes comme des vecteurs dans un espace à n dimensions, où chaque dimension représente un terme du vocabulaire d'indexation.

La représentation du document d, notée $\overrightarrow{d}$, est une application définie sur l'ensemble des termes du vocabulaire (noté T) telle que :

$$\forall\, t_i \in T,\ \overrightarrow{d}\ :\ t_i \rightarrow p_i$$

p_i est la pondération du terme t_i dans le document d. En général l'intervalle de valeurs pour p_i est $[0, 1]$. Ce choix arbitraire ne change rien au raisonnement. La représentation d'une requête est similaire à la représentation d'un document.

Dans ce modèle il existe plusieurs fonctions de pertinence pour comparer document et requête : produit scalaire, cosinus, mesure de *Dice*, mesure de *Jaccard*, etc. (cf. §2.2.4 de [13]). Le cosinus est très souvent utilisé comme mesure de pertinence, car il donne de bons résultats. Il estime l'angle entre deux vecteurs :

$$cos(\overrightarrow{d}, \overrightarrow{q}) = \frac{\overrightarrow{d} \cdot \overrightarrow{q}}{|\overrightarrow{d}| \times |\overrightarrow{q}|}$$

Cependant, deux problèmes importants apparaissent dans cette approche : le grand nombre de dimensions et l'indépendance des dimensions :

- le grand nombre de dimensions provient du fait que chaque terme du vocabulaire d'indexation devient une dimension de l'espace. Il faut donc étudier en détail la représentation des vecteurs sous peine de travailler avec des matrices creuses et très grandes ;
- les dimensions de l'espace sont orthogonales, ainsi, que les termes soient proches ou non "sémantiquement", ne change rien pour le modèle. Par exemple, « chat » et « félin » sont aussi indépendants que « chat » et « automobile » : $\cos(chat, felin) = \cos(chat, automobile) = 0$.

Un certain nombre de variantes de ce modèle ont été proposées : le modèle vectoriel généralisé, le modèle vectoriel sémantique, le modèle d'indexation sémantique latente, etc. (cf. [12, §2.7]). Une description du modèle vectoriel sémantique est proposée ci-dessous.

Le modèle vectoriel sémantique Le modèle vectoriel sémantique est une extension du modèle vectoriel classique. La principale différence vient de l'utilisation de concepts d'une ontologie plutôt que des mots-clés dans la représentation des documents et des requêtes. C'est ce modèle qui est utilisé dans les méthodes ExSID et ExSI^{2}D. Il faut noter que le problème de l'indépendance des dimensions n'est pas adressé par l'utilisation de concepts plutôt que de mots-clés.

Formellement, dans l'approche dite du modèle vectoriel sémantique, les dimensions de l'espace sont les concepts d'une ontologie. Le cadre général d'un tel modèle est celui de la figure 2.3 : requêtes et documents sont indexés sémantiquement et les documents sont classés par rapport à chaque requête en utilisant le cosinus comme mesure de pertinence.

Dans le modèle vectoriel sémantique, les concepts de l'ontologie sont souvent appelés les dimensions des vecteurs. Par exemple, considérons l'ontologie de la figure 2.4. Elle est composée de douze concepts, avec des liens de subsomption (*is-a*) entre eux. Par exemple le concept *public school* est une sorte (*is-a*) du concept *school*.

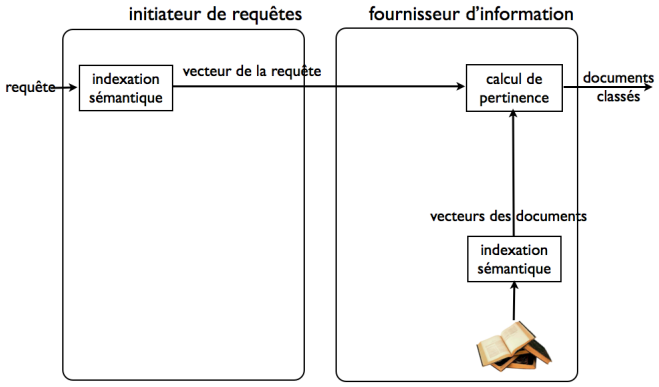

FIG. 2.3 – Cadre d'un système de recherche d'information utilisant le modèle vectoriel.

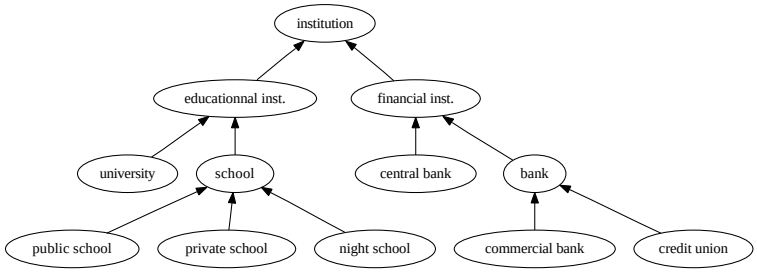

FIG. 2.4 – Une ontologie restreinte, composée de douze concepts avec les liens de subsomption (*is-a*).

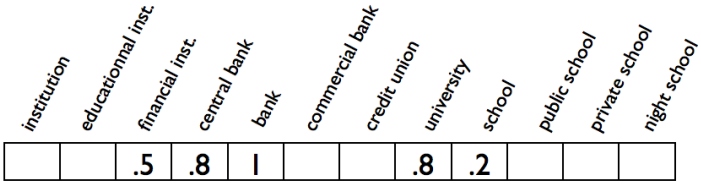

FIG. 2.5 – Le vecteur sémantique d'un document caractérisé sur l'ontologie de la figure 2.4.

La figure 2.5 présente un document d_i caractérisé par un vecteur sémantique dans l'espace défini par l'ontologie de la figure 2.4. Nous voyons que $\overrightarrow{d_i}[financial\ institution] = 0.5$, $\overrightarrow{d_i}[central\ bank] = 0.8$, $\overrightarrow{d_i}[bank] = 1$, $\overrightarrow{d_i}[university] = 0.8$ et $\overrightarrow{d_i}[school] = 0.2$. Ce qui signifie que pour le document est lié à ces cinq dimensions, plus fortement au concept *bank* qu'aux autres, même si les concepts *central bank* et *university* sont importants. En ce qui concerne les deux autres concepts, *financial institution* et *school*, leur pondération indique que ce ne sont pas des dimensions centrales pour le document.

2.2.3 Recherche d'information dans les réseaux P2P

Particularités de la RI dans les systèmes P2P

Dans un système P2P, les résultats d'un système de RI ne dépendent plus seulement de la manière dont sont indexés les documents et de la mesure utilisée pour comparer les documents avec les requêtes. En effet la répartition des données, la topologie du réseau et les algorithmes de routage utilisés pour faire transiter les requêtes vont avoir une grande influence dans les performances (pertinence des résultats, temps d'exécution et charge du réseau).

Les algorithmes de routage L'algorithme de routage choisi pour rechercher l'information dans un système P2P a une influence sur les résultats retournées par le système de RI. Par exemple, dans le cas des algorithmes Top-K (cf. section 2.1.2), il est évident que la valeur du TTL a une influence sur la qualité des résultats retournées par le système de RI (car tous les pairs du système ne reçoivent pas la requête). Plus le TTL est grand et plus le nombre de pairs recevant la requête est important : le nombre de documents pertinents retournés sera donc au moins égal sinon plus important. En effet on a :

$$\forall N > 0,\ D_{i+N} \geqslant D_i \Rightarrow P_{i+N} \geqslant P_i$$

où D_j est le nombre de documents accessibles avec un $TTL = j$ et P_j est le nombre de documents pertinents accessibles avec un $TTL = j$.

La stratégie choisie pour implémenter l'algorithme Top-K a une influence sur le temps d'exécution et sur la charge du réseau. En effet des optimisations pour diminuer le nombre de messages échangés permettent de diminuer la charge du réseau et de diminuer le temps d'éxécution.

La topologie du réseau La topologie du réseau a aussi un impact important sur les résultats. En effet, plus la connectivité des pairs (ie. le nombre de voisins directs que possède chaque pair) est importante et plus les pairs sont accessibles rapidement. En effet, le nombre maximum M_c de pairs accessibles avec un certain TTL est :

$$M_C = \sum_{i=0}^{TTL} C^i$$

Il est évident que plus la connectivité C est importante et plus le nombre maximum de pairs accessibles M_C est grand.

La répartition des données Dans un système P2P, les pairs contiennent des données qui leur sont propres. Selon la répartition des données dans le réseau, les résultats du système de RI seront plus ou moins bons. Par exemples, si les documents pertinents sont contenus dans les voisins directs du pair initiateur de la requête, alors les résultats seront bons, même avec un faible TTL. Par contre, si les documents pertinents sont très éloignés du pair initiateur, ils ne sont pas forcément accessibles (selon la valeur du TTL) : les résultats seront donc moins bons.

Évaluation d'un système de recherche d'information P2P

L'évaluation d'un système de recherche d'information P2P n'est pas un problème simple. Le fait que les données sont distribuées sur différents pairs, amène un certain nombre de problèmes. Le plus immédiat concerne le placement des données dans le réseau. En effet, pour évaluer un système de recherche d'information P2P, il faut être capable de placer les documents dans les pairs et de désigner les pairs initiateurs de requêtes, de manière réaliste.

L'évaluation des systèmes de RI centralisés est possible grâce à l'utilisation de corpus de test (cf. section 2.2.1). Jusqu'à présent, les corpus n'intègrent pas de données explicites permettant de gérer l'aspect distribué des systèmes de RI P2P.

Des solutions sont proposées pour exploiter les corpus de test "centralisés" dans le contexte P2P. Par exemple, Holz *et al.* [20] proposent de répartir les documents par auteur : tous les documents d'un auteur sont stockés dans un même pair. Cette approche est simple et permet d'obtenir une configuration réaliste ; entre autres parce qu'elle introduit de la réplication (un document écrit par plusieurs auteurs, apparaît dans plusieurs pairs). Le réseau de pairs peut être construit grâce aux références faites dans les documents : si un auteur cite un autre auteur, alors il est probable que ceux-ci soient proches ; autrement dit, il est probable qu'ils travaillent sur des sujets proches. Cette information peut être utilisée pour relier les pairs entre-eux. Bien entendu, pour que cette méthode fonctionne, il faut que la notion d'*auteur* soit présente dans le corpus. Cela n'est pas toujours le cas ; par exemple les corpus 20newsgroups, TREC-AP.

Neumann *et al.* [28] proposent un algorithme qui utilise les hyperliens entre les documents pour créer des clusters de documents. Comme les clusters générés sont disjoints, ils proposent aussi un algorithme pour créer "artificiellement" des overlaps (ie. introduire de la réplication de données dans le système). Il faut remarquer que cette méthode n'est pas non plus adaptable à tous les corpus car elle nécessite que les documents soient liés entre eux.

Witschel *et al.* [20] suggèrent de ne pas utiliser de corpus de test pour évaluer un système de RI P2P. Ceux-ci utilisent le fait que les résultats d'un système de RI P2P sont rarement meilleurs que ceux d'un système de RI centralisé, pour proposer de changer de référentiel : les résultats ne sont plus comparés à un jugement de pertinence humain, mais aux résultats d'un système de RI centralisé.

Les solutions proposées pour adresser les problèmes liés à l'évaluation de systèmes de RI P2P ont des limites. En particulier, les choix faits pour répartir les données, ou pour clusteriser les systèmes sont discutables. Ce problème reste donc très ouvert.

2.3 Interopérabilité sémantique

2.3.1 Hétérogénéité sémantique

Problématique

De manière générale, la conception d'une ontologie est une tâche complexe et critique car l'objectif est de représenter le « monde ». Il est raisonnable de penser que deux personnes peuvent avoir des points de vue différents sur le monde et sur la manière de le représenter. Par exemple, en Occident, la *vache* est considérée comme un animal domestique parfois destiné à être mangé ; alors qu'en Inde, elle est considérée comme un animal sacré. Il semble donc difficile d'imaginer qu'il est possible de concevoir une ontologie unanimement acceptée : l'existence d'une ontologie universelle est mise en cause.

Étant donné que nous considérons qu'il existe plusieurs ontologies (plusieurs représentations subjectives du monde), il semble alors raisonnable d'imaginer un système distribué où chaque participant peut fonctionner avec sa propre ontologie. Dans cette configuration, les pairs vont se retrouver confronté à un problème d'incompréhension avec leurs voisins : c'est ce qui s'appelle l'hétérogénéité sémantique. Le travail consistant à combler les différences entre les ontologies est appelé l'intégration sémantique. Il doit permettre la « communication »entre plusieurs ontologies. Pour cela, il est utile d'identifier les différences entre les ontologies.

Niveaux d'hétérogénéité

Quatre niveaux d'hétérogénéité sémantique peuvent être identifiés (cf. chapitre 2.1 de [16]) :

Le niveau syntaxique concerne essentiellement le fait qu'une ontologie peut être modélisée dans différents langages (OWL, RDF/RDFS, KIF). Pour tenter d'atténuer ce problème, des standards W3C tels que RDF/RDFS, OWL et SPARQL ont été proposés.

Le niveau terminologique concerne les différences de nommage des entités. Par exemple :

- deux entités identiques peuvent être nommées de manière différentes (polysémie, abbréviation) ;
- deux entités différentes peuvent être nommées de la même manière (synonymie) ;
- deux entités identiques peuvent être nommées de la même manière, mais dans des langues différentes ;

Le niveau conceptuel regroupe trois aspects : le point de vue duquel s'est placé le concepteur de l'ontologie (la perspective), le domaine du monde qu'il a cherché à représenter (la couverture), et le niveau de détail qu'il a voulu atteindre (la granularité). La perspective est issue de la difficulté à créer une ontologie avec un regard objectif.

Le niveau sémiotique intervient lors de l'utilisation des ontologies : un même concept peut être utilisé de plusieurs manière.

Maintenant que nous avons identifié les différents niveaux d'hétérogénéité, nous allons voir quelles sont méthodes permettant de repérer les correspondances entre ontologies.

2.3.2 Alignement d'ontologies

L'objectif du processus d'alignement d'ontologies est de trouver les "parties" d'ontologies sur lesquelles les participants s'accordent. Avant de voir comment se déroule ce processus (section 2.3.2), il faut d'abord définir ce qu'est un alignement.

Définition

Un alignement est un ensemble de correspondances exprimant les relations existantes entre deux ontologies. Ces relations sont accompagnées d'une valeur de confiance et peuvent être des relations d'équivalence, de spécialisation, de composition, etc. Un alignement doit pouvoir prendre en compte le niveau d'hétérogénéité conceptuel, c'est-à-dire la perspective, la granularité et la couverture.

Formellement, un alignement est un ensemble de 4-uplets (correspondences) : $\langle e_1, e_2, R, n\rangle$ tel que :
- e_1 et e_2 sont des entités (concepts, relations, etc.) de Ω_1 et Ω_2,
- R est une relation (équivalence, disjonction, spécialisation),
- n est un indice de confiance.

Processus d'alignement

En règle générale, un processus d'alignement fait intervenir plusieurs méthodes qui ont chacune pour objectif de trouver des correspondances entre deux ontologies. Ces méthodes peuvent être locales ou globales (cf. [24]). Nous allons, dans un premier temps, expliquer quelles sont ces méthodes, et nous verrons ensuite en quoi consiste le processus d'alignement.

Méthodes locales La méthode locale la plus basique utilise les chaînes de caractères pour trouver des correspondances. En fait, la recherche s'appuie sur les labels donnés aux entités. Pour que cela fonctionne bien, il faut commencer par normaliser les chaînes de caractères ; c'est-à-dire enlever les caractères spéciaux (apostrophes, ponctuation, etc.), mettre tous les caractères en minuscule, éliminer les mots inutiles (les stop-words). Il faut ensuite comparer les chaînes de caractères des entités considérées. Pour cela il faut disposer de mesures permettant de :
- déterminer si deux chaînes c_1 et c_2 sont égales,
- déterminer la proximité entre chaînes de caractères. Cela est utile pour déterminer si une chaîne c_1 est plus proche de c_2 ou de c_3. Les mesures de Hamming et Jaro permettent de faire cela.

Des méthodes plus avancées (de traitement du langage naturel) permettent d'utiliser la sémantique pour détecter les variations syntaxiques (ex : *cheval de trait* et *cheval de course et de trait*), morphologiques (ex : *cheval* et *chevaux*), sémantiques (ex : *étalon* et *cheval*) ou linguistiques (ex : *horse* et *cheval*). Encore une fois, il faut disposer de mesure permettant de :
- déterminer si deux termes sont des synonymes,
- déterminer la proximité entre deux termes.

Les mesures de synonymie et de Resnik permettent d'effectuer ces mesures.

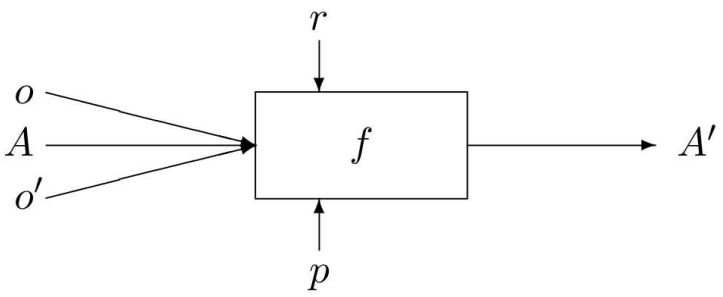

FIG. 2.6 – Processus d'alignement.

Les deux méthodes que nous venons de voir sont terminologiques. D'autres méthodes utilisent la structure des ontologies ou des entités pour trouver des correspondances : ce sont les méthodes structurelles externes ou internes. Les méthode structurelles internes utilisent le type et les contenus des entités alors que les méthodes structurelles externes utilisent les relations existantes entre les entités (hyperonymes, hyponymes, ancêtres, descendants, feuilles).

D'autres méthodes existent, parmi lesquelles les méthodes « extensionnelles »(elles utilisent les instances communes à deux ontologies pour trouver des correspondances) ou les méthodes sémantiques (elles utilisent des modèles théoriques sémantiques).

Méthodes globales Les méthodes globales utilisent parfois des techniques d'apprentissage en exploitant des bases de connaissances ou des statistiques calculées à partir d'alignement déjà effectués dont on connait déjà la qualité. Elles permettent aussi de prendre en compte une interaction avec un utlisateur : l'utilisateur peut déterminer si certaines correspondances ne sont pas correctes ou effectuer des choix lorsque la méthode ne permet pas de le faire.

L'intérêt principal des méthodes globales est qu'elles permettent d'itérer le processus jusqu'à trouver un point fixe (ie. on utilise des méthodes d'alignement en chaîne jusqu'à ce que l'itétation n'améliore plus l'alignement).

Processus d'alignement On peut dire que le processus d'alignement permet de combiner plusieurs méthodes d'alignements (locales ou globales). La figure 2.6 présente une étape du processus d'alignement où o et o' sont les ontologies à aligner, A est un alignement déjà existant (ne contenant éventuellement aucune correspondance), r des ressources (par exemple l'interaction d'un utilisateur ou des bases de connaissances) et p des paramètres (par exemple un seuil minimal de confiance qu'on peut accorder à une correspondance). Le résultat A' est un alignement qui peut lui-même devenir un alignement d'entrée pour une nouvelle itération.

2.3.3 ExSI^{2}D

Les alignement d'ontologies permettent d'exploiter les parties communes entre différentes ontologies mais le nombre de correspondances peut s'avérer trop faible pour obtenir de bons résultats, en particulier

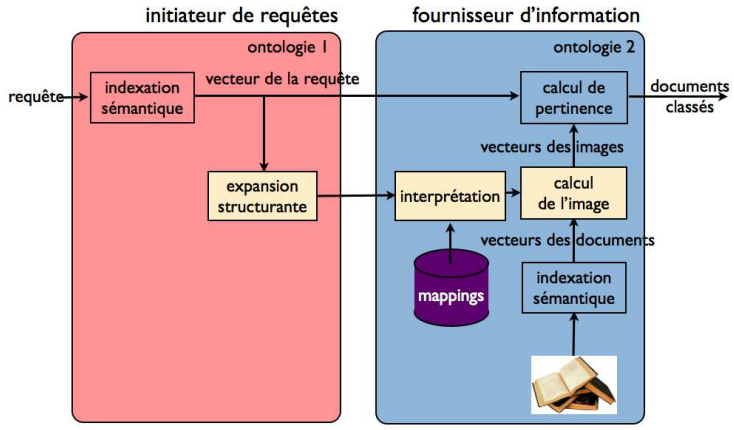

FIG. 2.7 – La méthode ExSI^{2}D est composée de trois modules : expansion structurante de requêtes, calcul d'image de documents et interprétation.

en RI car si les requêtes ne sont pas comprises, alors les fournisseurs d'information ne peuvent pas répondre convenablement. Ventresque *et al.* proposent une méthode permettant d'exploiter les parties partagées (connues grâce à un alignement), et les parties non-partagées : ExSI^{2}D. Comme l'illustre la figure 2.7, cette méthode est composée des modules d'expansion structurante de requêtes, de calcul d'images de documents et d'interprétation. Ces trois modules sont présentés dans les sections ci-dessous.

Expansion structurante des requêtes

Expansion classique L'indépendance des dimensions des vecteurs peut-être pénalisante dans le processus de recherche d'information. Prenons un exemple concret pour illustrer cela.

Soient d_1 et d_2 des documents représentés par les vecteurs $\overrightarrow{d_1}$ et $\overrightarrow{d_2}$ et q une requête représentée par $\overrightarrow{q}$. Supposons que nous utilisons l'ontologie de la figure 2.4 et que $\overrightarrow{d_1}$ et $\overrightarrow{q}$ sont seulement pondérés pour le concept $bank$ et que $\overrightarrow{d_2}$ est pondéré uniquement sur le concept $commercial\ bank$. Dans ce cas, d_2 n'apparaît pas comme pertinent par rapport à q (car $cos(\overrightarrow{d_2}, \overrightarrow{q}) = 0$) alors que d_1 est pertinent. Pourtant il n'est pas déraisonnable de penser que l'initiateur de la requête peut être interessé par le document d_2 (car le concept $commercial\ bank$ *is-a* $bank$).

Une des solutions proposée pour répondre à ce problème est l'enrichissement de requêtes (ou expansion de requête). Bellot [29] présente différentes méthodes d'enrichissement de requêtes. Dans l'exemple précédent, l'enrichissement de requête produirait une requête q' pondérée sur le concept $bank$ (comme dans q) et sur le concept $commercial\ bank$ car $bank$ et $commercial\ bank$ sont sémantiquement proches.

Il faut noter que si l'enrichissement permet de trouver des documents pertinents qui ne l'auraient pas été sans expansion, il est aussi fréquemment source de bruit. Autrement dit, l'expansion de requête donne parfois de l'importance à des concepts qui ne sont pas représentatifs du besoin exprimé par l'utilisateur. D'autre part, l'enrichissement étant fait au sein d'un seul et même vecteur, il n'est pas possible de déterminer si le concept c pondéré dans q' était pondéré initialement ou si sa pondération est issue de l'expansion (ou bien des deux).

Expansion structurante Ventresque *et al.* [34] proposent une expansion de requête où chaque propagation de concept est mémoriser dans des vecteurs séparés : des dimensions sémantiquement enrichies (DSE). L'ensemble de ces vecteurs constitue l'*expansion structurante de requête*. La requête initiale n'étant pas modifiée, elle peut être utilisé lors du processus de classement par pertinence des documents. La propagation de l'intérêt d'un concept est obtenue par composition d'une fonction de similarité et d'une fonction de propagation et ne s'effectue que sur les concepts importants de la requête.

Similarité La *fonction de similarité* sert à quantifier la similarité de deux concepts c et c' appartenant à une ontologie Ω. Plusieurs approches existent pour calculer la similarité entre deux concepts mais dans tous les cas nous pouvons dire que :

- $\forall c' \in \Omega, sim_c(c') \in [0,1]$,
- $sim_c(c) = 1$,
- $sim_c(c_1) \geqslant sim_c(c_2) \Rightarrow c_1$ est plus similaire à c que c_2.

La mesure de similarité se base parfois sur la structure globale de l'ontologie : c'est le cas des mesures de Wu&Palmer ([35]) et de Bidault ([14]) ; et d'autres se basent également sur le contenu informationnel des concepts : mesure de Resnik ([30]) ou de Jiang-Conrath ([22]).

Propagation La valeur quantifiant le lien entre un concept c' et un concept c de la requête q, s'obtient par la formule :

$$v \times Pf(sim_c(c'))$$

où v est la pondération de c dans q et Pf est une fonction de propagation telle que :

- $\forall x \in [0,1], Pf(x) \in [0,1]$,
- $Pf(1) = 1$,
- $\forall \alpha, \beta \in [0,1], \alpha \leqslant \beta \Rightarrow Pf(\alpha) \leqslant Pf(\beta)$.

La figure 2.8 présente un exemple de fonction de propagation où le concept considéré est $bank$ (ce concept est pondéré par 1 dans la requête). Les concepts de l'ontologie sont positionnés selon la valeur de la fonction sim_{bank}. Ainsi le concept $commercial\ bank$ obtient la pondération $1 \times Pf(sim_{bank}(commercial\ bank)) = 0.6$ dans la DSE dont le concept central est $bank$ (voir figure 2.9).

Les expansions classiques (cf. section 2.3.3) sont effectuées par les fournisseurs d'information alors que le module d'expansion structurante est utilisé par l'initiateur de la requête. Cela paraît logique dans le sens où il est le mieux placé pour étoffer sa requête en fonction de ses connaissances.

La figure 2.9 présente un exemple d'une requête étendue : elle est composée de la requête elle même ($query$) et de deux dimensions sémantiquement enrichies ($SED_{university}$ et SED_{bank}).

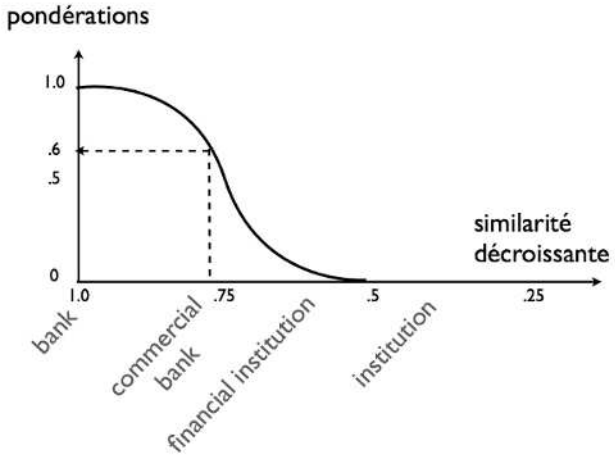

FIG. 2.8 – Exemple d'une fonction de propagation. Le concept considéré est $bank$.

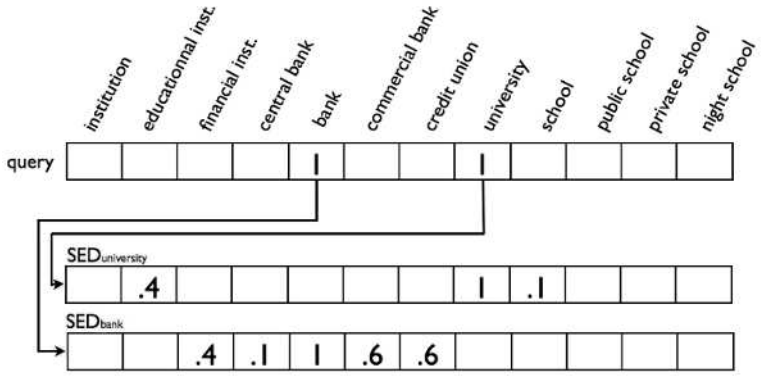

FIG. 2.9 – L'ensemble des concepts centraux de la requête créent un ensemble de dimensions sémantiquement enrichies.

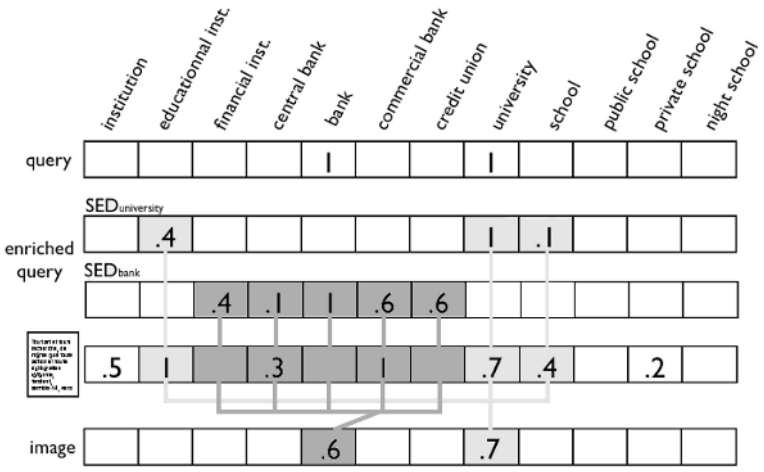

FIG. 2.10 – Une requête enrichie, un document et son image après traitement des deux DSEs de la requête enrichie.

Image d'un document au travers d'une requête

L'image d'un document (notée $\overrightarrow{i_d}$) est la représentation du document d adaptée à la requête q et est calculée du côté du fournisseur. Le résultat de la construction de l'image sert à donner une valeur de pertinence au document par rapport à la requête initiale.

Supposons que nous avons un document et une requête étendue, comme dans la figure 2.9. Le calcul de l'image d'un document s'effectue en deux étapes : prise en compte des DSEs et recopie du document.

Prise en compte des DSEs Dans cette étape, l'objectif est de pondérer les concepts centraux des DSEs dans $\overrightarrow{i_d}$: dans notre exemple, il s'agit de pondérer $\overrightarrow{i_d}[university]$ et $\overrightarrow{i_d}[bank]$. La pondération doit prendre en compte la pondération des concepts proches du concept central. Pour cela, les dimensions pondérées d'une DSE sont repliées sur le concept central de cette DSE dans l'image du document. Le calcul se fait de la manière suivante :

$$\overrightarrow{i_d}[c] := max(\overrightarrow{d}[c'] \times \overrightarrow{DSE_c}[c']),\ \forall c' \in \Omega$$

En pratique, il suffit de considérer les concepts c' dont la pondération n'est pas nulle dans la DSE. Ainsi nous avons : $\overrightarrow{i_d}[university] := 0.7$ et $\overrightarrow{i_d}[bank] := 0.6$. À ce stade du processus nous sommes dans la situation présentée dans la figure 2.10.

Recopie des dimensions non représentées du document Dans cette étape, il s'agit de pondérer les concepts qui n'ont pas été pris en compte dans l'étape précédente. En fait, il suffit de considérer l'ensemble des concepts c tels que :

- c est pondéré dans $\overrightarrow{d}$ et
- c n'est pondéré dans aucune DSE.

Pour chaque concept c vérifiant ces conditions, nous avons $\overrightarrow{i_d}[c] := \overrightarrow{d}[c]$.

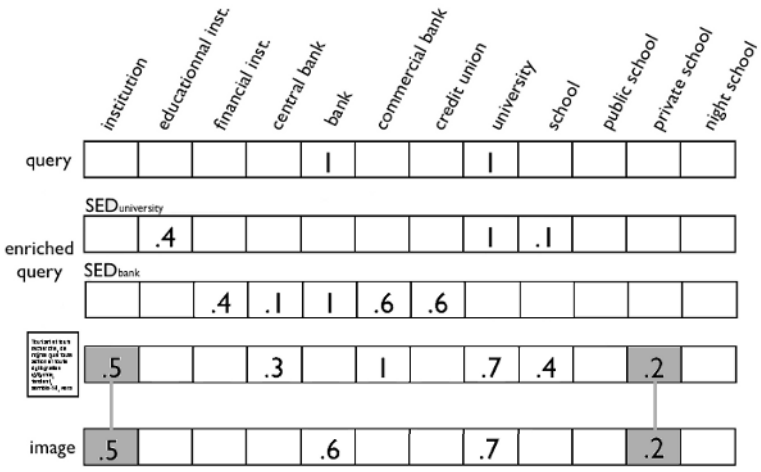

FIG. 2.11 – Une requête enrichie, un document et son image à la fin du traitement.

De cette manière nous obtenons $\overrightarrow{i_d}[institution] := 0.5$ et $\overrightarrow{i_d}[private\ school] := 0.2$. À ce stade du processus nous sommes dans la situation présentée dans la figure 2.11.

Interprétation

Quand l'initiateur de la requête et le fournisseur des documents utilisent des ontologies différentes, des problèmes d'incompréhension peuvent survenir. Ce problème est d'autant plus présent dans le cas de systèmes P2P où le nombre de fournisseurs (et donc d'ontologies utilisées) peut être très important.

Ce problème est en partie adressé par le matching d'ontologies. En effet, étant données deux ontologies Ω_1 et Ω_2, le matching d'ontologies nous permet de connaître l'ensemble des correspondances (c_1, c_2) telles que $c_1 \in \Omega_1$ et $c_2 \in \Omega_2$. L'initiateur de la requête et le fournisseur peuvent ainsi se comprendre. Cela fonctionne bien lorsque tous les concepts de la requête sont compris par le fournisseur ; c'est-à-dire que pour tout concept c_q de la requête, il existe une correspondance entre c_q et $c_2 \in \Omega_2$.

Le problème persiste lorsque l'initiateur de la requête utilise un concept c_q qui n'est pas compris par le fournisseur de documents : Ventresque *et al.* suggèrent d'interpréter la requête. L'interprétation produit des dimensions sémantiquement interprétées (DSI) dans l'espace vectoriel définit par Ω_2. À chaque DSE décrite dans Ω_1 correspond une DSI décrite dans Ω_2.

La création d'une DSI se fait en deux étapes : choix du concept central et pondération des concepts.

Choix du concept central Si le concept central de la DSE est partagé par les deux utilisateurs, alors le concept central de la DSI est celui de la DSE. Dans le cas contraire, il faut chercher un candidat dans Ω_2 pour devenir le concept central de la DSI. Cette recherche se fait en comparant l'allure des fonctions d'interprétation avec l'allure des fonctions de propagations. Une explication plus précise est donnée au paragraphe 5.2.1 de [34].

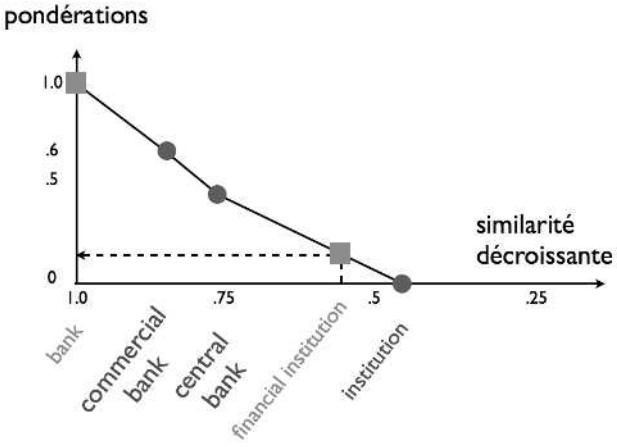

FIG. 2.12 – Pondération des concepts non partagés (les carrés) $bank$ et $financial\ institution$ grâce à fonction d'interprétation.

Pondération des concepts Il s'agit de pondérer chaque concept c de l'ontologie Ω_2 dans la DSI. Pour ce faire nous pouvons considérer deux cas de figure :

- si c a un équivalent dans Ω_1, alors c conserve, dans la DSI, le poids qu'il avait dans la DSE.
- si c n'a pas d'équivalent dans Ω_1, alors le poids de c dans la DSI est calculé grâce à la fonction d'interprétation.

Dans la figure 2.12, nous pouvons voir que le fournisseur de documents a classé ses concepts grâce au concept central $bank$. Les concepts partagés (les ronds) conservent leurs poids. La fonction d'interprétation permet d'attribuer des poids aux concepts non partagés (les carrés).

L'interprétation permet d'utiliser des concepts non partagés pour caractériser des documents par rapport à une requête.

Chapitre 3

Mesures d'interopérabilité sémantique

3.1 Cadre général

Comme nous l'avons vu dans la section 2.3.1, l'hétérogénéité est un problème difficile à adresser. Les alignements résolvent en partie le problème, et la méthode ExSI^{2}D permet d'améliorer l'interopérabilité entre les participants. Néanmoins, nous ne disposons pas de mesures de similarité ou de proximité sémantique entre participants. Celles-ci pourraient être très utiles, en particulier dans les systèmes P2P où le nombre de participants est important. Dans ce cas précis, des mesures d'interopérabilité nous permettraient de placer les pairs de façon optimisée (ie. les regrouper dans des zones où la compréhension est bonne) et de router les informations (par exemple les requêtes) de manière à ce qu'elles soient bien comprises.

Plus que de mesurer l'hétérogénéité, notre objectif est mesurer la capacité de deux participants à interopérer : nous appelons cela le degré d'interopérabilité. Pour cela, nous considérons que chaque participant utilise une ontologie et qu'un processus d'alignement nous permet de disposer d'un alignement de qualité ; c'est à dire que toutes les correspondances existantes sont détectées et qu'aucune fausse correspondance n'est détectée.

David et Euzenat ([18]) présentent et comparent différentes mesures de distance entre deux ontologies. En fait, ils recherchent des mesures permettant de calculer rapidement la distance entre deux ontologies avant de les aligner. Dans notre cas, c'est l'inverse : nous disposons de l'alignement et nous voulons déterminer à quel point l'alignement nous permet de communiquer facilement.

3.2 Préléminaires

3.2.1 Mesures recherchées

En général, les mesures proposées pour quantifier la similarité entre ontologies sont des mesures de distance : elles respectent donc les trois propriétés suivantes :

1. la symétrie,
2. la séparation,
3. l'inégalité triangulaire.

Ce que nous recherchons n'est pas une distance car nous considérons que l'interopérabilité entre deux ontologies n'est pas symétrique : Ω_1 peut être parfaitement interopérable avec Ω_2 sans que l'inverse soit vrai (si Ω_1 est incluse dans Ω_2).

Nous cherchons des mesures ayant la signature suivante :

$$m : \left| \begin{array}{lcl} E \times E & \mapsto & [0,1] \\ \Omega_1, \Omega_2 & \rightarrow & m(\Omega_1, \Omega_2) \end{array} \right.$$

et ayant les caractéristiques suivantes :

- m est réflexive ($m(\Omega_1, \Omega_1) = 1$),
- $m(\Omega_1, \Omega_2) = 1$ signifie que Ω_1 et Ω_2 peuvent interopérer de manière parfaite.
- $m(\Omega_1, \Omega_2) = 0$ signifie que Ω_1 et Ω_2 ne peuvent pas interopérer.

3.2.2 Rappel et notations

Un alignement entre deux ontologies Ω_1 et Ω_2 est un ensemble de correspondance, où chaque correspondance peut être vue comme un 4-uplet $\langle c_1, c_2, R, n \rangle$ où c_1 et c_2 sont des concepts de Ω_1 et Ω_2, R est une relation (d'équivalence, de généralisation, etc.) entre c_1 et c_2, et n est une mesure de confiance. Dans un premier temps, nous choisissons de n'utiliser que les relations d'équivalence entre concepts.

Notons C_{Ω_1}, l'ensemble des concepts de l'ontologie Ω_1 et A_{Ω_1} l'ensemble des concepts de Ω_1 partagés avec Ω_2.

3.2.3 Mesure d'importance d'un concept

Pour mesurer l'interopérabilité entre deux ontologies nous allons utiliser l'alignement qui les lie. Cet alignement nous permet de connaitre les concepts qui sont partagés et ceux qui ne le sont pas. Un concept non partagé dégrade l'interopérabilité mais il faut remarquer ques tous les concepts non partagés n'ont pas le même impact. Par exemple, pour un utilisateur spécialisé dans la physique, il est plus dommagable de ne pas partager le concept « atome » que le concept « arbre ». Il faut donc considérer que nous disposons d'une mesure d'importance sur les concepts d'une ontologie. Elle est définie ainsi.

$$i_\Omega : \left| \begin{array}{lcl} C_\Omega & \mapsto & [0,1] \\ c & \rightarrow & i_\Omega(c) \end{array} \right.$$

Dans l'exemple que nous venons de présenter, $i_\Omega(atome) > i_\Omega(arbre)$.

On peut imaginer plusieurs stratégies pour déterminer cette mesure. La plus simple est de considérer que tous les concepts ont la même importance. Dans ce cas on a :

$$\forall c \in C_\Omega, i_\Omega(c) = 1$$

Nous aurions alors : $i_\Omega(atome) = i_\Omega(arbre) = 1$

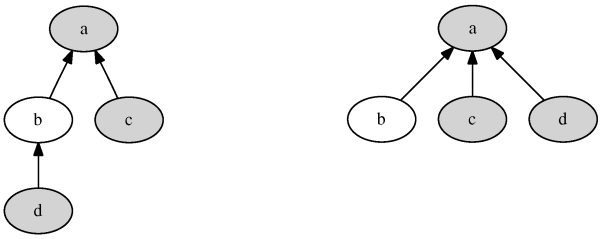

FIG. 3.1 – Exemple de deux ontologies : Ω_1 (à gauche) et Ω_2 (à droite). Les nœuds représentent les concepts et les arcs représentent les liens de subsomption.

On peut aussi utiliser une mesure de similarité définie pour les concepts de Ω. On peut définir $i_\Omega(c)$ ainsi :

$$\forall c \in C_\Omega, i_\Omega(c) = sim_{racine}(c)$$

où $racine$ est l'élément le plus général de l'ontologie. Dans ce cas, nous aurions : $i_\Omega(atome) > i_\Omega(arbre)$ si $sim_{racine}(atome) > sim_{racine}(arbre)$ ($i_\Omega(atome) \leqslant i_\Omega(arbre)$ sinon).

Enfin, le contexte global peut aussi servir à donner une valeur d'importance aux concepts. Par exemple, dans le cas de la RI, on peut considérer que l'importance d'un concept dépend de sa fréquence d'apparition dans les index. Ainsi, les concepts les plus importants sont ceux qui sont le plus souvent utilisés dans les documents considérés.

D'autres stratégies peuvent évidemment être utilisées et les mesures proposées ci-dessus ne sont que des exemples.

3.3 Contribution

Dans cette section, nous proposons trois mesures d'interopérabilité. Pour chacune d'entre elles nous donnons une explication et sa définition théorique. Nous allons également appliquer ces mesures sur un exemple et éventuellement entamer une discution sur leurs limites.

Pour les exemples, nous utilisons les ontologies simples de la figure 3.1. Le scenario lié aux exemples est le suivant : nous disposons de deux ontologies (Ω_1 et Ω_2) et nous voulons savoir laquelle est la plus interopérable avec une troisième ontologie (Ω_3). Nous ne connaissons pas la structure de cette dernière mais les alignements A_{Ω_1} et A_{Ω_2} nous permettent de savoir quels concepts sont partagés avec Ω_3 (les concepts grisés de la figure 3.1), et quels concepts ne le sont pas (les concepts blanc de la figure 3.1).

Pour simplifier les calculs, nous considérons, dans nos exemples, que les concepts ont tous la même importance. Cela donne :

$$\forall c \in \Omega_1, i_{\Omega_1}(c) = 1 \text{ et } \forall c \in \Omega_2, i_{\Omega_2}(c) = 1$$

3.3.1 Mesure de compréhensibilité

En se basant uniquement sur la connaissance des concepts partagés, nous pouvons définir une première mesure :

$$m_1(\Omega_x, \Omega_y) = \frac{\sum_{c \in A_{\Omega_x}} i_{\Omega_x}(c)}{\sum_{c \in C_{\Omega_x}} i_{\Omega_x(c)}}$$

Cette mesure définit un degré d'interopérabilité entre deux ontologies en ne tenant compte que de l'importance des concepts partagés par rapport à l'ensemble des concepts de la première ontologie : elle ne tient pas compte de l'organisation des concepts entre eux.

On vérifie évidemment que $m_1(\Omega_x, \Omega_y) \in [0, 1]$, que si aucun concept n'est partagé nous avons $m_1(\Omega_x, \Omega_y) = 0$; et enfin que si tous les concepts sont partagés nous avons $m_1(\Omega_x, \Omega_y) = 1$.

Exemple Il porte sur les ontologies de la figure 3.1. Comme les ontologies Ω_1 et Ω_2 partagent autant de concepts avec Ω_3 (3) et qu'elles sont toutes deux composées de 4 concepts, nous avons :

$$m_1(\Omega_1, \Omega_3) = m_1(\Omega_2, \Omega_3) = \frac{3}{4} = 0,75$$

Discussion La mesure m_1 permet de dire que les ontologies Ω_1 et Ω_2 sont interopérables de la même manière avec Ω_3. Pourtant nous pouvons nous attendre à ce que les degrés d'interopérabilité soient différents car le concept non partagé (b) a plus de voisins partagés avec Ω_3 dans Ω_1 (a et d) que dans Ω_2 (seulement a). Pour prendre en compte cette notion, nous proposons la mesure m_2 qui utilise les relations entre les concepts.

3.3.2 Mesure de densité

Dans l'ontologie Ω_1 de la figure 3.1, le concept b n'est pas partagé mais on peut tenter d'en comprendre le sens grâce à ses voisins. La notion de voisinnage est donnée par les fonctions de similarité utilisées par les pairs. Par exemple, considérons que sim^1 et sim^2 sont les fonctions de similarité utilisées respectivement sur Ω_1 et Ω_2, et que nous avons les mesures présentées sur la figure 3.2.

Les ontologies Ω_1 et Ω_3 ne partagent pas le concept b mais ils partagent les concepts d et a qui sont relativement proches de b (similarité $> 0,5$). Le sens de b pourra donc être compris grâce à a et d. Par contre, dans Ω_2, seul le concept a est proche de b. Il sera donc plus difficile de comprendre le sens de b dans Ω_2. On constate que la densité de concepts partagés autour d'un concept a un impact sur l'interopérabilité : la mesure m_2 vise à utiliser cette notion de densité.

On définit la densité autour d'un concept c de la manière suivante :

$$densite_{\Omega_x} : \left| \begin{array}{lcl} C_{\Omega_x} & \mapsto & [0,1] \\ c & \rightarrow & densite_{\Omega_x}(c) \end{array} \right.$$

où

$$densite_{\Omega_x} = \begin{cases} \frac{\sum_{c' \in A_{\Omega_x}} i_{\Omega_x}(c') \times sim_c^x(c')}{\sum_{c' \in C_{\Omega_x}} i_{\Omega_x}(c') \times sim_c^x(c')} & si\ c \notin A_{\Omega_x} \\ 1 & sinon \end{cases}$$

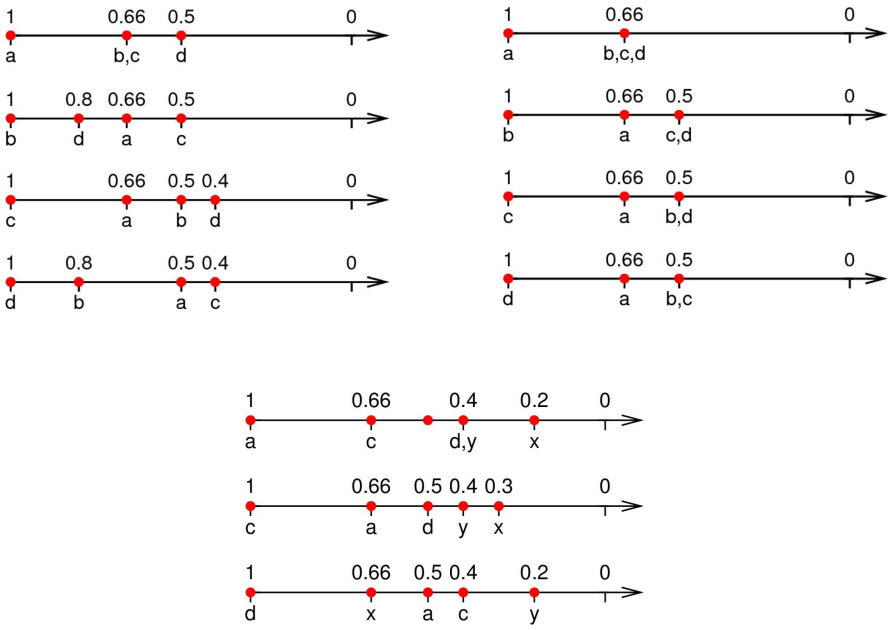

FIG. 3.2 – Résultats des fonctions de similarité sim^1 (en haut à gauche), sim^2 (en haut à droite) et sim^3 (en bas). Les valeurs pour sim^1 et sim^2 sont calculées avec la mesure de Wu&Palmer. Nous supposons que les valeurs pour sim^3 nous sont données (peu importe la manière dont elles ont été calculées).

Nous pouvons maintenant définir le degré d'interopérabilité ainsi :

$$m_2(\Omega_x, \Omega_y) = \frac{\sum_{c \in C_{\Omega_x}} densite_{\Omega_x}(c)}{\|C_{\Omega_x}\|}$$

L'idée de cette mesure est que plus l'entourage d'un concept est dense, et plus il sera possible d'en comprendre le sens, même s'il n'est pas partagé.

Nous vérifions que $m_2(\Omega_x, \Omega_y) \in [0, 1]$, que $m_2(\Omega_x, \Omega_y) = 0$ si aucun concept n'est partagé et que $m_2(\Omega_x, \Omega_y) = 1$ si tous les concepts sont partagés.

Exemple Nous considérons encore les ontologies de la figure 3.1 et les mesures de similarité présentées sur la figure 3.2 pour calculer $m_2(\Omega_1, \Omega_3)$ et $m_2(\Omega_2, \Omega_3)$.

$$m_2(\Omega_1, \Omega_3) = \frac{densite_{\Omega_1}(a) + densite_{\Omega_1}(b) + densite_{\Omega_1}(c) + densite_{\Omega_1}(d)}{4}$$

Comme a, c et d sont partagés avec Ω_3, leurs densités sont égales à 1. La densité autour de b se calcule ainsi :

$$\begin{aligned} densite_{\Omega_1}(b) &= \frac{sim^1_b(a) + sim^1_b(c) + sim^1_b(d)}{sim^1_b(a) + sim^1_b(b) + sim^1_b(c) + sim^1_b(d)} \\ &= \frac{0,66 + 0,5 + 0,8}{0,66 + 1 + 0,5 + 0,8} = 0,66 \end{aligned}$$

Cela donne :

$$m_2(\Omega_1, \Omega_3) = \frac{1 + 0,66 + 1 + 1}{4} = 0,915$$

De la même manière on trouve :

$$densite_{\Omega_2}(a) = densite_{\Omega_2}(c) = densite_{\Omega_2}(d) = 1$$

et

$$densite_{\Omega_2}(b) = \frac{sim_b^2(a) + sim_b^2(c) + sim_b^2(d)}{sim_b^2(a) + sim_b^2(b) + sim_b^2(c) + sim_b^2(d)} = 0,62$$

et donc :

$$m_2(\Omega_2, \Omega_3) = \frac{1 + 0,62 + 1 + 1}{4} = 0,905$$

Finalement on trouve que $m_2(\Omega_1, \Omega_3)$ est légèrement meilleur que $m_2(\Omega_2, \Omega_3)$. Le résultat de cette mesure prend bien en compte le fait que la densité à un impact sur l'interopérabilité.

Discussion L'utilisation de cette mesure considère, de manière implicite, que la fonction de similarité est la même pour les deux participants. Cette hypothèse est forte et dans le cas où elle n'est pas respectée, la mesure m_2 perd sa légitimité. En effet pour expliquer un concept à un interlocuteur, les participants vont utiliser les concepts qui lui sont proches sémantiquement. La notion de proximité dépend de la fonction de similarité donc si l'un considère que les concepts proches d'un concept c sont ses hypernymes, et que l'autre considère que les concepts proches sont les hyponymes, alors il va y avoir une incompréhension. La mesure suivante tente de pallier ce problème.

3.3.3 Mesure de désordre

Pour prendre en compte le fait que deux participants peuvent utiliser des fonctions de similarités différentes, il faut utiliser des mesures de désordre. Il faut distinguer le cas où la mesure de désordre porte sur un concept partagé, et le cas où la mesure de désordre porte sur un concept non-partagé.

Dans tous les cas, on a :

$$desordre_{\Omega_x,\Omega_y} : \left| \begin{array}{lll} C_{\Omega_x} & \mapsto & [0,1] \\ c & \rightarrow & desordre_{\Omega_x,\Omega_y}(c) \end{array} \right.$$

Avant de présenter les mesures de désordre, il faut introduire la fonction f définie ainsi :

$$f : \left| \begin{array}{lll} [0,1] & \mapsto & [0,1] \\ x & \rightarrow & f(x) \end{array} \right.$$

Cette fonction permet de donner plus d'importance aux concepts proches de c et moins d'importance à ceux qui sont éloignés. Elle peut être :

- l'identité : $f_1(x) = x$. Elle permet de donner une importance proportionnelle à la similarité avec le concept c.

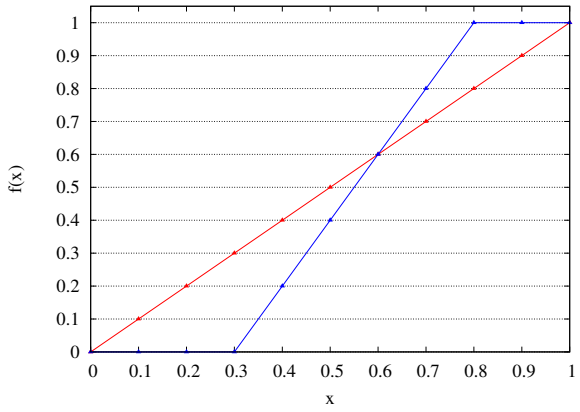

FIG. 3.3 – Exemple de fonctions f : f_1 (en rouge) et f_2 (en bleu).

– linéaire par morceaux :

$$f_2(x) = \left| \begin{array}{ll} 0 & si\ x \in [0, \alpha] \\ \frac{x-\alpha}{\beta-\alpha} & si\ x \in]\alpha, \beta[\\ 1 & si\ x \in [\beta, \alpha] \end{array} \right.$$

Dans ce cas, elle permet de donner beaucoup d'importance aux concepts très proches de c (ceux dont la similarité est supérieure à β), aucune importance aux concepts qui sont trop éloignés de c (ceux dont la similarité est inférieure à α), et des valeurs intermédiaires aux concepts dont la similarité est entre α et β.

– constante : $f_3(x) = \gamma$ où $c \in [0, 1]$. Elle donne une importance équivalente ($\forall \gamma \in [0, 1]$) à tous les concepts quelque soit leur similarité avec c).

La figure 3.3 présente deux fonctions : f_1 et f_2 (avec $\alpha = 0.3$ et $\beta = 0.8$).

Concept partagé Dans le cas d'un concept partagé, la mesure de désordre détermine à quel point l'organisation des concepts partagés autour du concept c est semblable dans Ω_x et dans Ω_y ? Si $desordre_{\Omega_x,\Omega_y}(c)$ est égal à :

– 0, cela signifie que les concepts partagés, sont ordonnés de la même façon dans Ω_x (l'ordre étant donné par la fonction de similarité sim_c^x) et dans Ω_y (avec sim_c^y).
– 1, cela signifie que les concepts partagés, sont ordonnés de la manière tout à fait différente dans Ω_x (avec sim_c^x) et dans Ω_y (avec sim_y^2).

On propose la mesure suivante :

$$desordre_{\Omega_x,\Omega_y}(c) = \frac{\sum_{c' \in A_{\Omega_x}} f(sim_c^x(c')) \times diff_{\Omega_x,\Omega_y}(c, c')}{\sum_{c' \in A_{\Omega_x}} f(sim_c^x(c'))}$$

La fonction $diff_{\Omega_x,\Omega_y}$ est définie ainsi :

$$diff_{\Omega_x,\Omega_y} : \left| \begin{array}{lll} A_{\Omega_x} \times A_{\Omega_x} & \mapsto & [0, 1] \\ c, c' & \rightarrow & diff_{\Omega_x,\Omega_y}(c, c') \end{array} \right.$$

avec

$$diff_{\Omega_x,\Omega_y}(c,c') = \frac{|rang_c(c',\Omega_x) - rang_c(c',\Omega_y)|}{\|A_{\Omega_x}\|}$$

La fonction $rang_c(c',\Omega_i)$ donne le rang du concept c' dans le classement des concepts partagés classés grâce à la fonction sim_c^i. Par exemple, sur la figure 3.2, nous avons $rang_b(d) = 2$. Notons que $rang_a(b) = rang_a(c) = 2$.

Concept non-partagé Dans le cas d'un concept de Ω_x non-partagé, il n'est pas possible de connaître l'ordre de ses voisins dans Ω_y (justement parce qu'il n'est pas partagé entre Ω_x et Ω_y). Il faut donc définir une fonction $desordre'_{\Omega_x,\Omega_y}$. Elle est définie ainsi :

$$desordre'_{\Omega_x,\Omega_y} : \left| \begin{array}{lcl} C_{\Omega_x} \setminus A_{\Omega_x} & \mapsto & [0,1] \\ c & \rightarrow & desordre'_{\Omega_x,\Omega_y}(c) \end{array} \right.$$

avec

$$desordre'_{\Omega_x,\Omega_y}(c) = \frac{\sum_{c' \in C_{\Omega_x}} f(sim_c^x(c')) \times g_{\Omega_x,\Omega_y}(c')}{\sum_{c' \in C_{\Omega_x}} f(sim_c^x(c'))}$$

Encore une fois, la fonction f permet de donner plus d'importance aux concepts proches de c et moins d'importance à ceux qui sont éloignés.

La fonction $g_{\Omega_x,\Omega_y}(c)$ vaut :
- 1 quand c n'est pas partagé ($c \notin A_{\Omega_x}$),
- $desordre_{\Omega_x,\Omega_y}(c)$ quand c est partagé ($c \in A_{\Omega_x}$).

Mesure d'interopérabilité Pour mesure l'interopérabilité en tenant compte du désordre des concepts partagés et non-partagés, nous pouvons définir la mesure m_3 ainsi :

$$m_3(\Omega_x,\Omega_y) = 1 - \frac{\sum_{c \in A_{\Omega_x}} desordre_{\Omega_x,\Omega_y}(c) + \sum_{c \notin A_{\Omega_x}} desordre'_{\Omega_x,\Omega_y}(c)}{\|C_{\Omega_x}\|}$$

On vérifie que $m_3(\Omega_x,\Omega_y) \in [0,1]$ et que $m_3(\Omega_x,\Omega_y) = 0$ si aucun concept n'est partagé. Par contre il faut noter que même si tous les concepts sont partagés, nous n'avons pas nécessairement $m_3(\Omega_x,\Omega_y) = 1$ car cette mesure prend en compte le désordre de chaque concept. Pour avoir $m_3(\Omega_x,\Omega_y) = 1$, il faut que tous les concepts soient partagés et que leurs proximités les uns par rapports aux autres soient identiques dans Ω_x et Ω_y.

Exemple Les applications de cette mesure se basent sur le scenario présenté en introduction (cf. figures 3.1 et 3.2).

$$m_3(\Omega_1,\Omega_3) = 1 - \frac{desordre_{\Omega_1,\Omega_3}(a) + desordre'_{\Omega_1,\Omega_3}(b) + desordre_{\Omega_1,\Omega_3}(c) + desordre_{\Omega_1,\Omega_3}(d)}{4}$$

Il faut calculer les mesures de désordre :

$$\begin{aligned}
desordre_{\Omega_1,\Omega_3}(a) &= \frac{diff_{\Omega_1,\Omega_3}(a,a) + diff_{\Omega_1,\Omega_3}(a,c) + diff_{\Omega_1,\Omega_3}(a,d)}{3} = \frac{0+0+0}{3} = 0 \\
desordre_{\Omega_1,\Omega_3}(c) &= \frac{diff_{\Omega_1,\Omega_3}(c,a) + diff_{\Omega_1,\Omega_3}(c,c) + diff_{\Omega_1,\Omega_3}(c,d)}{3} = \frac{0+0+0}{3} = 0 \\
desordre_{\Omega_1,\Omega_3}(d) &= \frac{diff_{\Omega_1,\Omega_3}(d,a) + diff_{\Omega_1,\Omega_3}(d,c) + diff_{\Omega_1,\Omega_3}(d,d)}{3} = \frac{0+0+0}{3} = 0 \\
desordre'_{\Omega_1,\Omega_3}(b) &= \frac{desordre_{\Omega_1,\Omega_3}(a) + 1 + desordre_{\Omega_1,\Omega_3}(c) + desordre_{\Omega_1,\Omega_3}(d)}{4} = 0,25
\end{aligned}$$

On obtient finalement :

$$m_3(\Omega_1,\Omega_3) = 1 - \frac{0+0,25+0+0}{4} = 0,94.$$

Nous pouvons calculer $m_3(\Omega_2,\Omega_3)$ de la même manière et nous trouvons :

$$\begin{aligned}
m_3(\Omega_2,\Omega_3) &= 1 - \frac{desordre_{\Omega_2,\Omega_3}(a) + desordre'_{\Omega_2,\Omega_3}(b) + desordre_{\Omega_2,\Omega_3}(c) + desordre_{\Omega_2,\Omega_3}(d)}{4} \\
&= 1 - \frac{0,11+0,27+0+0}{4} = 0,90
\end{aligned}$$

Nous remarquons que :

$$m_3(\Omega_1,\Omega_3) > m_3(\Omega_2,\Omega_3)$$

Cela est normal aux vues des valeurs des fonctions de similarités (cf. figure 3.2). Ce résultat exprime le fait que l'explication des concepts non-partagés faite par Ω_1 sera mieux comprise par Ω_3 que l'explication de Ω_2.

3.4 Bilan

Les mesures que nous venons de proposer permettent de mesurer l'interopérabilité entre deux pairs utilisant des ontologies. Elles mesurent chacune des choses différentes et sont complémentaires. Grâce à ces mesures nous allons pouvons améliorer optimiser un certain nombre d'algorithmes (le routage de requêtes par exemple).

Ces mesures n'ont pas encore été implémentées mais cela fait partie des travaux à venir. Nous pourrons alors tester ces mesures en situation réelle ; c'est-à-dire sur de grosses ontologies (WordNet, Sensus, etc.).

Chapitre 4

Mise en place d'ExSI^{2}D

Notre objectif est la mise en place de la méthode ExSI^{2}D dans un système P2P hétérogène. Les mesures que nous venons de proposer vont nous servir à mesurer l'interopérabilité entre les pairs étant donnée l'hétérogénéité. Avant de pouvoir expérimenter la méthode, il a fallu réaliser un certain nombre de tâches de modélisation et de développement. La section 4.1 présente le travail réalisé : modélisation des stuctures de données pour la sémantique, amélioration de l'outils PeerSim, et implémentation de la méthode ExSI^{2}D. La section 4.2 présente les expérimentations que nous avons effectuées (scenarii, paramètres de simulations, résultats) et la section 4.3 conclut cette partie et ouvre des perspectives.

4.1 Modélisation et développement

4.1.1 Sémantique

Le modèle de RI que nous utilisons est le modèle vectoriel (cf. section 2.2.2) : il faut donc analyser précisément ce qu'est un vecteur sémantique pour pouvoir le représenter correctement.

Vecteur sémantique basique

Un vecteur sémantique est un ensemble de dimensions (représentant les concepts) associées à des valeurs (les pondérations). Nous pouvons voir un vecteur sémantique comme un tableau associatif de taille N où les clés sont des concepts, et les valeurs des pondérations. Comme nous voulons avoir des vecteurs normalisés à 1, nous devons considérer que les pondérations sont des réels compris entre 0 et 1. Par ailleurs, le nom (ou l'url) de l'ontologie à laquelle appartiennent les concepts, est également représentée sous forme de chaîne de caractère dans les vecteurs sémantiques. Le type des concepts doit être générique pour pouvoir définir réutiliser la structure avec n'importe quel type de concepts : des chaînes de caractères, des entiers (comme dans WordNet), etc.

Mis à part les accesseurs classiques, les vecteurs sémantiques doivent disposer d'opérations de base, comme par exemple :

- le cosinus : étant donné que c'est la mesure utilisée pour calculer la proximité entre un vecteur et un autre est le cosinus, il est indispensable de modéliser cette opération.
- la multiplication : lors du calcul d'image d'un document par rapport à une requête, il est nécessaire de faire le produit de deux vecteurs ; il faut une opération capable d'effectuer le produit.

– la recherche du concept pondéré le plus fortement : cette méthode sera aussi utile lors du calcul d'image.

Nous disposons maintenant d'une structure permettant de représenter le résultat de l'indexation d'un document ou d'une requête (cf. classe $basic.SemanticVector$ de la figure 4.1), mais cela n'est pas tout à fait suffisant puisque l'indexation est un processus pouvant être long et coûteux que nous ne voulons pas répéter plusieurs fois. Il est donc nécessaire de permettre le stockage des vecteurs sémantiques. Nous avons choisi de stocker les vecteurs dans des fichiers XML car ce format permet de représenter clairement leur structure, en particulier avec la définition d'une DTD (Document Type Definition). Par ailleurs, l'API Jdom ([5]) facilite la lecture et l'écriture dans de tels fichiers : la manipulation sera donc très simple. Pour faciliter davantage la manipulation (lecture et écriture) des vecteurs, nous avons créé des classes ($Reader$ et $Writer$) qui permettent de lire ou d'écrire plusieurs vecteurs à la fois.

Vecteur sémantique identifiable

Les vecteurs sémantiques servent à stocker le résultat de l'indexation de documents ou de requêtes. Après indexation, les vecteurs basiques ne permettent pas de retrouver la ressource qu'ils représentent. Pour résoudre ce problème, nous avons spécialisé le type *vecteur sémantique basique* (basic.SemanticVector). Les *vecteurs sémantiques identifiables* permettent de référencer la ressource qu'ils représentent. Le type de l'identifiant est générique pour pouvoir utiliser aussi bien une url (chaîne de caractères), un entier (numéro unique dans un corpus de test), etc. Cette structure (identifiable.SemanticVector) est aussi stockable dans un fichier XML.

Vecteur sémantique localisable

Notre objetif est de placer les vecteurs sémantiques dans un système P2P. Il faut donc être capable de déterminer le pair dans lequel sera placé un vecteur. Afin de permettre de faire cela statiquement, nous avons spécialisé le type *vecteur sémantique identifiable* afin d'ajouter un attribut $location$. Cet attribut est également générique pour pouvoir utiliser n'importe quel type de donnée : identifiant du pair (entier), url du pair (chaîne de caractères), etc. Cet attribut permet, en particulier, de déterminer si deux vecteurs doivent être placés dans le même pair.

Des classes de lecture et d'écriture sont également proposées pour permettre de stocker les « vecteurs sémantiques localisables ». Contrairement aux autres $Readers$, celui proposé à ce niveau regroupe les vecteurs en fonction de leurs localisations, afin de pouvoir récupérer plus facilement et plus efficacement les données devant être placées dans le même pair.

Implémentation

Les vecteurs sémantiques présentés ci-dessus ont été implémentés en Java. Afin de respecter certaines normes de qualité logicielle, les classes et les méthodes ont été annotées avec des commentaires Javadoc. Ainsi le code est plus compréhensible et maintenable plus facilement. Des tests unitaires ont aussi été mis en place pour tester les méthodes critiques. Ils ont été réalisés avec Junit.

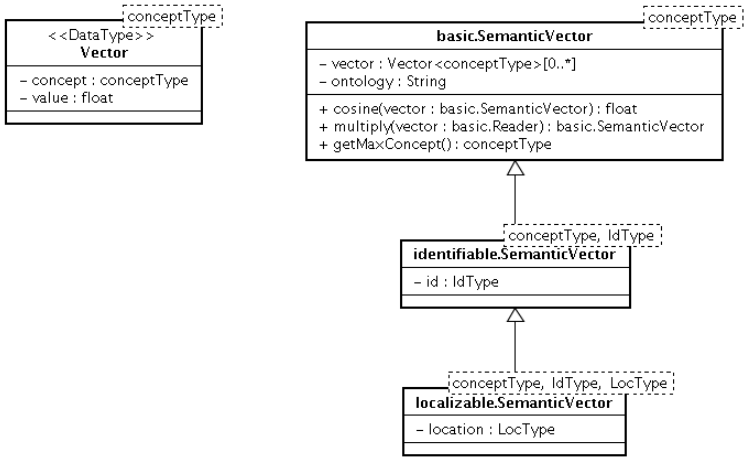

FIG. 4.1 – Diagramme de classes présentant les différents vecteurs sémantiques : basique, identifiable et localisable.

4.1.2 Algorithmes de routage dans PeerSim

L'existant

Un ensemble d'algorithmes, programmés en Java, ont été proposés par Philippe LAMARRE et William DEDZOÉ. Ces algorithmes permettent le routage de requêtes top-K et la récupération de réponses dans un réseau P2P simulé avec PeerSim (cf. section 2.1.3). Le scénario testé avec ces algorithmes est le suivant : les pairs contiennent une liste d'entiers ; et un pair envoie une requête contenant un entier afin d'obtenir les K entiers les plus proches de sa requête. Dans ce scénario, la requête contient un entier, les pairs contiennent des entiers et les réponses sont elles-mêmes des entiers.

Objectif

Nous voudrions pouvoir envoyer des requêtes et récupérer les résultats en considérant que les types des données, de la requête et des réponses ne sont pas liés. Par exemple, les pairs peuvent contenir des tableaux d'entiers, la requête peut être un nombre réel et les résultats peuvent représenter la chaîne de caractères identifiant le tableau dont la moyenne des éléments est proche de la requête. Sur cet exemple il est évident que le type des requêtes, des données et des réponses sont indépendants : il faut paramétrer les classes existantes pour rendre le code générique.

Contribution

Le travail a consisté à rendre génériques une quarantaine de classes. La partie gauche de la figure 4.2 montre des classes telles qu'elles étaient avant le refactoring et la partie droite montre les classes modifiées. Toutes les classes qui ont fait l'objet d'une modification ont été commentées pour améliorer la lisibilité et facilité la maintenance. Ce travail a été l'occasion de comprendre le fonctionnement des algorithmes proposés.

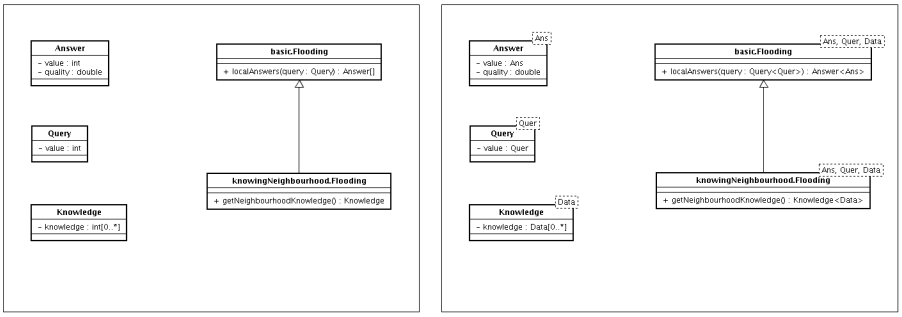

FIG. 4.2 – Exemple de quelques classes de PeerSim : version orginale à gauche et version générique à droite.

4.1.3 ExSI^{2}D

L'un des objectifs de ce stage est la mise en place de la méthode ExSI^{2}D dans un système P2P. Cette méthode avait été testée dans un cadre centralisé où seulement deux participants interagissent : l'un émet des requêtes et l'autre fournit des documents en réponse aux requêtes reçues (cf. [34]). Avant de pouvoir l'évaluer dans un système distribué, un certain nombre de modifications s'imposent ; autant du point de vue modélisation que du point de vue optimisation. En effet, dans le cadre centralisé, la méthode ExSI^{2}D, mettait plusieurs jours à traiter 225 requêtes sur un ensemble de 1400 documents.

Contribution

Modélisation des structures de données La méthode ExSI^{2}D utilise des DSE (Dimension Sémantiquement Enrichie) pour expliquer les concepts prenant part à une requête émise par un participant. En fait, chaque requête est accompagnée, par ce que nous avons appelé une requête expliquée (*explained query*). Nous avons créé une structure de donnée, appelé vecteur expliqué, constitué :

- du vecteur lui-même,
- des explications de chaque concept du vecteur.

L'explication d'un concept (*concept explanation*) est un vecteur permettant d'expliquer ce que signifie le concept. La figure 2.9 (page 28) montre un vecteur pondérant deux concepts ($university$ et $bank$) et deux explications de concepts (une pour chaque concept) : le tout forme un vecteur expliqué.

Nous avons modélisé ces structures ($explainedVector$ et $conceptExplanation$) à l'aide de nos propres structures : $basic.SemanticVector$, $identifiable.SemanticVector$ et $localizable.SemanticVector$. Ces structures vont être utilisées par les services que nous présentons dans la section suivante.

Modélisation de l'architecture La méthode ExSI^{2}D est composée de trois modules principaux : expansion (ou explication), image de document et interprétation. Le but est de pouvoir utiliser chacun de ces modules de manière indépendante. Les services web semblent répondre à ce besoin.

La figure 4.3 présente les différents modules que nous avons identifiés, ainsi que les dépendances existantes entre eux. Nous voyons par exemple que le module de propagation utilise le module de similarité. Celui-ci peut calculer la similarité en utilisant différentes mesures (Wu&Palmer, Bidault, etc.) et cela ne

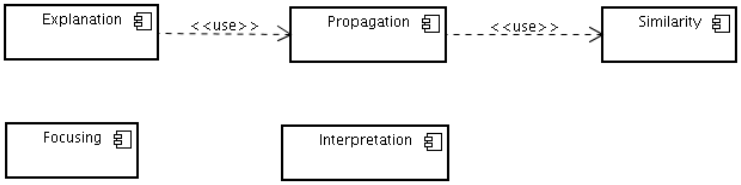

FIG. 4.3 – Modules composant l'architecture d'ExSI^{2}D : similarité, propagation, explication, image de document et interpération.

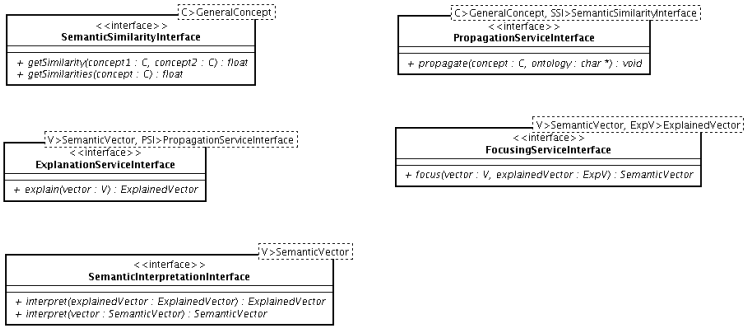

FIG. 4.4 – Services proposés par ExSI^{2}D sous forme d'interfaces.

fait pas de différence pour le module de propagation : il appelle le service et récupère les résultats de la même manière. Il en va de même pour les autres modules ; par exemple le module d'explication qui utilise le module de propagation. La figure 4.4 présente les différents services proposés par l'application ExSI^{2}D.

Pour mettre les modules sous forme de services, nous avons utilisé des interfaces Java paramétrées. Il est ensuite possible d'appeler les services en instanciant correctement les paramètres.

Optimisations

Les optimisations ont principalement été réalisées par Anthony VENTRESQUE. Elles se basent essentiellement sur deux choses : la mise en cache des calculs effectués et la numérotation des concepts de WordNet.

Mise en cache des résultats Lorsqu'on cherche à classer tous les concepts par ordre de similarité (par rapport à un concept c donné), il faut faire le calcul de similarité $\|C_\Omega\| - 1$ fois ($\|C_\Omega\|$ étant le nombre de concepts contenus dans Ω). Cela devient très coûteux, surtout quand la taille de l'ontologie est importante. En plus, nous pouvons avoir besoin de ce classement à de nombreuses reprises (sur chaque pair du système par exemple). La solution proposée est donc de stocker les résultats la première fois qu'ils sont calculés. Les fois suivantes, il suffit de consulter les fichiers contenant les données.

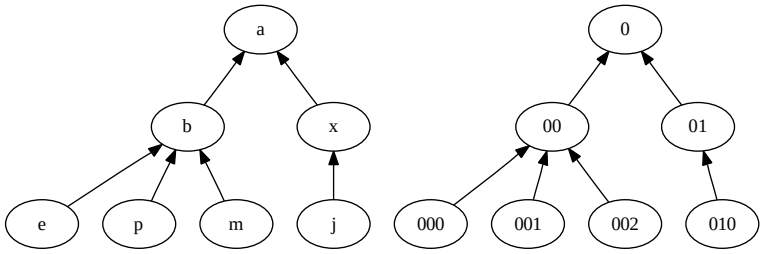

FIG. 4.5 – Exemple d'une ontologie composée de sept concepts reliés par des liens de subsomption (à gauche) et exemple de renumérotation des concepts (à droite).

Numérotation des concepts Lors du calcul de similarité, nous avons souvent besoin de rechercher l'ancêtre commun le plus proche de deux concepts. Cette tâche peut être coûteuse, en particulier si le nombre d'ancêtres est important. La solution mise en place repose sur une numérotation particulière des concepts de l'ontologie. En fait, il suffit de numéroter la racine de l'ontologie avec 0. Ensuite chacun de ses enfants seront numérotés à partir de 0 : 00, 01, 02, etc. La figure 4.5 montre un exemple de numérotation sur une petite ontologie. Pour rechercher l'ancêtre commun le plus proche de deux concepts, il suffit de prendre la partie commune maximale de leur numérotation : on trouve ainsi le numéro du concept recherché. Sur l'exemple de la figure 4.5, on retrouve rapidement que l'ancêtre commun le plus proche de m et x est a car la partie commune de leurs numérotations est 0 (ce qui est le numéro du concept a).

4.2 Expérimentations

4.2.1 Objectifs

Dans cette section, nous allons présenter les expérimentations que nous avons effectuées. L'objectif principal de ces expérimentations est de comparer deux méthodes de RI dans le cadre hétérogène :

- une méthode basique qui se contente de comparer les documents et les requêtes avec le cosinus,
- la méthode ExSI^{2}D.

En fait, l'objectif est d'observer le comportement des méthodes en fonction du degré d'hétérogénéité (cf. section 4.2.2).

Dans un premier temps, nous voulons vérifier que nous obtenons bien les mêmes résultats que ceux présentés dans la thèse d'Anthony VENTRESQUE ([34]) : c'est ce qui est présenté dans la section 4.2.3. Dans un second temps, nous avons comparé les deux méthodes dans un système P2P simulé avec PeerSim : c'est ce que nous présentons dans la section 4.2.4.

Avant d'aller plus loin, nous allons exposer les conditions dans lesquelles les expérimentations ont été réalisées.

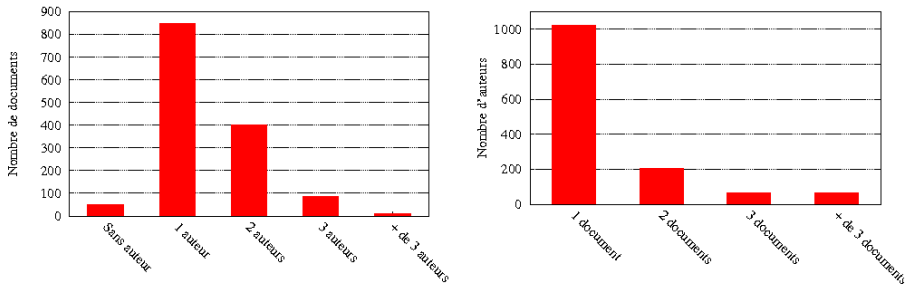

FIG. 4.6 – Nombre de documents par auteur (à gauche) et nombre d'auteurs par documents (à droite).

4.2.2 Paramètres de simulation

Corpus de test : Cranfield

Les méthodes que nous voulons tester sont spécifiques à la RI donc il nous faut utiliser un corpus de test pour quantifier la qualité de nos résultats. Le corpus de Cranfield est un bon candidat car il avait déjà été utilisé pour les expérimentations dans [34] et que le nombre de requête est relativement important.

Le corpus de Cranfield contient 225 requêtes et 1400 documents ayant été écrits par 1356 auteurs différents. Comme le montre la figure 4.6, certains auteurs ont écrit plusieurs documents, certains documents ont été écrits par plusieurs auteurs et certains documents n'ont pas d'auteur.

Pour pouvoir manipuler les documents plus facilement, nous avons effectué une modification. Le corpus de Cranfield est formatté en XML et au préalable, nous avions une balise XML $author$ contenant un ou plusieurs auteurs. La séparation entre les différents auteurs était faite par une virgule ou le terme « and ». Afin que l'extraction des auteurs puissent être faite automatiquement, nous avons créé une balise $authors$ contenant une ou plusieurs balise $author$. Les balises $author$ ne contiennent désormais plus qu'un seul auteur.

Indexation

L'indexation des documents et des requêtes a été effectuée grâce au services web développés par un groupe de Master 1 (projet Mysin, [27]). Les services mis à disposition permettent l'indexation sémantique de documents. L'indexeur sémantique utilisé est Riio. Il utilise lui-même un indexeur lexical : Lucène ([7]).

Gestion de l'interopérabilité

Comme nous l'avons vu dans la section 2.3.1, l'hétérogénéité vient du fait que les participants utilisent différentes ontologies. L'interopérabilité est possible grâce aux alignements d'ontologies. Dans un soucis de simplicité, nous avons choisi de n'utiliser qu'une seule ontologie : WordNet. Le choix de cette ontologie a été motivé par son volume : nombre important de concepts ($\simeq$72000), de relations, de types de relation, etc. De plus l'API JWNL (Java WordNet Library, [6]) permet d'utiliser WordNet simplement.

Comme nous utilisons une seule ontologie, l'interopérabilité est parfaite. Nous avons dû dégrader l'interopérabilité artificiellement. Pour cela, nous avons "fabriqué" nous-mêmes les alignements entre ontologies. Nous supposons que nous disposons de dix ontologies différentes ($wordNet_0$, $wordNet_1$, etc.). Il suffit alors de stocker des alignements : pour chaque couple d'ontologies $\langle wordNet_i, wordNet_j \rangle$ nous avons une liste mélangée $l_{i,j}$ contenant tous les concepts de WordNet. Avec les expérimentations, nous voulons observer l'évolution de la qualité des systèmes de RI face à la dégradation de l'interopérabilité. Pour mesurer l'interopérabilité, nous avons utilisé la mesure m_1 que nous présentons dans la section 3.3. Nous avons considéré que les concepts ont tous la même importance : $i_\Omega(c) = 1, \forall c \in \Omega$. Ainsi nous avons :

$$m_1(\Omega_x, \Omega_y) = \frac{\sum_{c \in A_{\Omega_x}} i_{\Omega_x}(c)}{\sum_{c \in C_{\Omega_x}} i_{\Omega_x}(c)} = \frac{\|A_{\Omega_j}\|}{\|C_{\Omega_j}\|}$$

Pour obtenir un degré d'interopérabilité de X%, il suffit de considérer que le participant utilisant Ω_x ($wordNet_x$) a seulement conscience qu'il partage X% des concepts avec le participant utilisant Ω_k ($wordNet_k$) en considérant que l'alignement ne contient que les X% premiers concepts présents dans la liste $l_{x,y}$.

Pour être réaliste, nous avons fait en sorte que les listes non-ordonnées $l_{i,j}$ soient différentes pour tous couples d'ontologies $(wordNet_i, wordNet_j)$. De plus, nous considérons que l'alignement n'est pas symétrique : l'alignement entre Ω_i et Ω_j n'est pas le même qu'entre Ω_j et Ω_i. Comme dans notre cas nous n'exploitons que les liens d'équivalence des ontologies, cette hypothèse est un peu forte (car la relation d'équivalence est symétrique).

La figure 4.7 montre que le nombre de mapping supprimés dans les requêtes varie linéairement avec le nombre de mapping supprimés. Les mesures sont faites sur l'ensemble des concepts des requêtes en considérant la moyenne sur tous les couples d'ontologies possibles. On vérifie ainsi, qu'en moyenne, le degré d'hétérogénéité entre ontologies se repercute bien sur les requêtes.

Dans nos simulations nous allons faire varier le nombre de mappings supprimés afin d'observer l'évolution de la qualité des systèmes de RI face à la variation du degré d'interopérabilité. Le degré d'interopérabilité évolue de cette manière : 0%, 10%, ... 100%.

Les valeurs de précision et de rappel obtenues avec la méthode basique (celle utilisant uniquement le cosinus comme mesure de pertinence) et dans le cadre centralisé et homogène, seront utilisées comme valeurs de référence. Nous faisons cela pour obtenir des graphes plus lisibles et pour que les résultats ne semblent pas dépendre de l'indexation (qui a un impact très fort sur la qualité d'un système de RI).

4.2.3 Cadre centralisé

Scénario

Dans ce scénario, nous avons deux participants :

- un initiateur de requêtes : il envoie les 225 requêtes du corpus les unes après les autres et attend les résultats.
- un fournisseur d'information : il possède l'ensemble des documents (1400) et lorsqu'il reçoit une requête, recherche dans la liste des documents ceux qui y répondent le mieux.

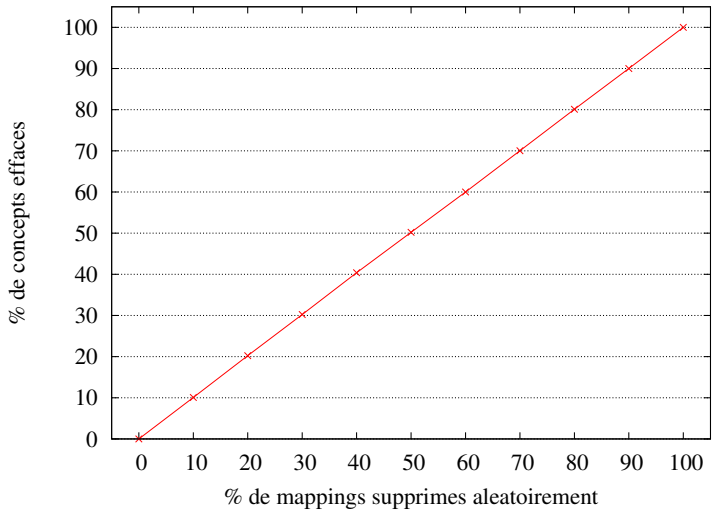

FIG. 4.7 – Pourcentage de concepts effacés pour l'ensemble des requêtes de Cranfield en fonction du pourcentage de mapping supprimés.

Interprétation des résultats

Nous pouvons voir sur la figure 4.8 que la méthode ExSI^{2}D obtient de meilleurs résultats que le cosinus : cela confirme les résultats présentés dans la thèse d'Anthony VENTRESQUE ([34]). Nous remarquons néanmoins plusieurs différences avec les résultats obtenus dans [34].

Premièrement, nous remarquons que les résulats que nous avons obtenus pour le cosinus sont assez bons (au dessus de la droite $f(x) = x$). Nous pensons que cela est dû au fait que l'absence de certains concepts de la requête peut améliorer les résultas. Dans ce cas, ces concepts peuvent être considérés comme du bruit pour la requête.

Deuxièmement, nous n'observons pas le même palier entre 70 et 100% d'interopérabilité que dans les résultats d'A. VENTRESQUE. Ce phénomène est issu du fait que l'interprétation n'est pas capable de retrouver tous les concepts non partagés même avec un fort degré d'interopérabilité. Nous avons remarqué que cela se produit lorsque plusieurs concepts frères non-partagés sont des feuilles (ie. qu'ils n'ont pas d'hyponymes dans WordNet). L'indexation a pondéré un grand nombre de concepts feuilles : il paraît donc logique que l'interprétation ne puisse pas toujours retrouver tous les concepts (même en utilisant une seule ontologie). L'indexation utilisée lors des expérimentations d'A. VENTRESQUE ponderait certainement moins de tels concepts.

Les résultats que nous avons obtenus sont proches de ceux que nous attendions même si nous observons quelques nuances. Nous allons voir dans la section 4.2.4 ce qu'il en est dans un cadre distribué.

4.2.4 Cadre distribué

Les simulations dans un cadre distribué amènent de nouvelles problématiques : placement des données, routage des requêtes et des réponses, organisation des pairs, etc.

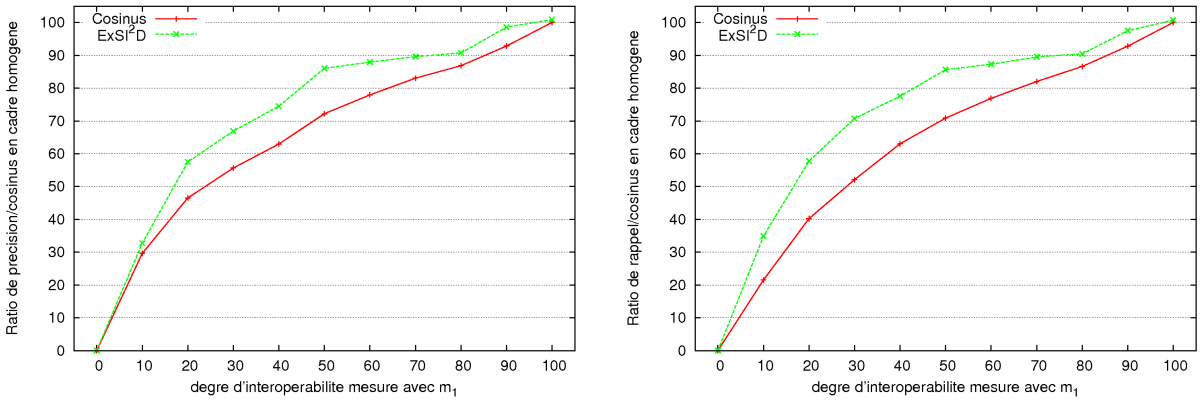

FIG. 4.8 – Évolution, dans un cadre centralisé, du ration de précision (à gauche) et de rappel (à droite) entre les deux méthodes (cosinus et ExSI^{2}D) et le cosinus dans le cadre homogène, en fonction du degré d'interopérabilité.

Topologie du système et placement des données

Pour fixer la topologie du système P2P et la répartition des données dans le système, nous avons étudié deux cas.

Premier cas d'étude Dans ce cas d'étude, nous proposons de placer les documents dans les pairs selon l'auteur (comme le suggèrent Holz *et al.* dans [20]) : chaque pair stocke les documents d'un auteur. Comme certains documents n'ont pas d'auteur, nous les plaçons dans des pairs différents. Nous avons donc un système contenant 1408 pairs, car nous avons 1356 auteurs et 52 documents sans auteurs.

Une telle configuration permet de travailler des un système relativement grand (1400 pairs) mais la répartition des données fait qu'il n'y a, en général, qu'un seul document par pair : cela ne semble pas très réaliste. Par ailleurs le temps d'exécution des algorithmes est très long car le nombre de pair est important. Cette topologie ne nous permet pas de répéter les expérimentations.

Second cas d'étude Pour éviter les inconvénients rencontrés dans le premier cas d'étude, nous proposons une deuxième solution. Comme précédemment nous proposons de répartir les documents selon les auteurs. Par contre, au lieu d'attribuer un auteur par pair, nous faisons en sorte de répartir les auteurs de façon à n'avoir que 100 pairs dans le système. En moyenne chaque pair est donc responsable des documents de dix auteurs différents. Cette configuration permet d'exécuter les méthodes de RI plus rapidement et de considérer un système où chaque pair contient plusieurs documents ; par contre la taille du système est moins importante.

Discussion Holz *et al.* proposent de lier les pairs entre eux en utilisant les auteurs des documents. Ainsi, deux auteurs ayant participé à l'écriture d'un même document seraient voisins. Pour le moment, nous n'avons pas choisi cette stratégie car le placement des pairs ne va pas influencer les résultats. En effet, nous allons inonder le système avec les requêtes : tous les pairs accessibles sont atteints car la valeur du TTL le permet (cf. §4.2.4).

Dans les deux cas d'étude, le degré de connectivité des pairs (nombre de voisins par pair) est fixé à 4. Cette valeur semble réaliste, car lors de l'étude effectuée sur le réseau Gnutella, Ripeanu *et al.* ont constaté que le degré moyen de connectivité est 3,4 (cf. [31]).

Gestion de l'interopérabilité

Dans le cadre centralisé, la gestion de l'interopérabilité était relativement simple : il suffisait de choisir une ontologie pour chacun des deux pairs et de faire varier le nombre de correspondances entre ces deux ontologies. Dans le cadre distribué, le nombre de participants est beaucoup plus important. Il est nécessaire de répondre aux questions suivantes :

- Combien faut-il utiliser d'ontologies différentes étant donnée la topologie du système ?
- Comment doit-on répartir les ontologies dans le système ? Faut-il les regrouper par zone (clusterisation) ?
- Comment faut-il gérer le degré d'interopérabilité dans le réseau ?

Chacune des ces questions soulève des problèmes complexes qui nécessitent une étude approfondie. Étant donné qu'elles ne font pas l'objet central de ce stage, nous avons tenté d'y répondre en justifiant nos choix.

Il est difficile d'imaginer que tous les participants d'un système distribué utilisent des ontologies différentes car la création (ou la modification) d'ontologies est une tâche difficile. Nous avons choisi d'utiliser 10 ontologies ; nous pensons que c'est assez pour simuler correctement l'hétérogénéité sémantique, et que cette valeur permet d'effectuer les expérimentations dans de bonnes conditions (espace disque, mémoire). Il serait bien sûr intéressant de faire varier le nombre d'ontologies pour observer l'impact qu'il a sur les résultats des systèmes de RI (cf. section 4.3).

En ce qui concerne la répartition dans le système, nous avons fait le choix de répartir les ontologies de manière aléatoire pour ne pas être placé dans un cas particulier ; par exemple en créant des zones où le degré d'interopérabilité est à 100% (c'est de la clusterisation). De plus, le regroupement de pairs en fonction de l'ontologie qu'ils utilisent, fait partie des améliorations que nous pensons mettre en place, entre autre grâce aux mesures que nous avons proposé dans la section 3.

Pour gérer l'interopérabilité entre les pairs, il y a plusieurs solutions. En particulier, pour fixer l'interopérabilité du système à X%, il est possible de :

- fixer à X% le degré d'interopérabilité entre tous les pairs : c'est une répartition uniforme de l'hétérogénéité ;
- fixer des degrés d'interopérabilité différents entre chaques couples de pairs et faire en sorte que la moyenne des degrés soit égale à X%.

Nous avons choisi la première solution car elle est simple à mettre en place et qu'il parait plus logique de commencer par celle-ci. Encore une fois, il sera intéressant de faire varier cette répartition afin de nous approcher d'un cas vraiment réaliste.

Algorithme de routage

Nous avons choisi d'utiliser l'algorithme Fully Distributed présenté dans la section 2.1.2 pour plusieurs raisons :

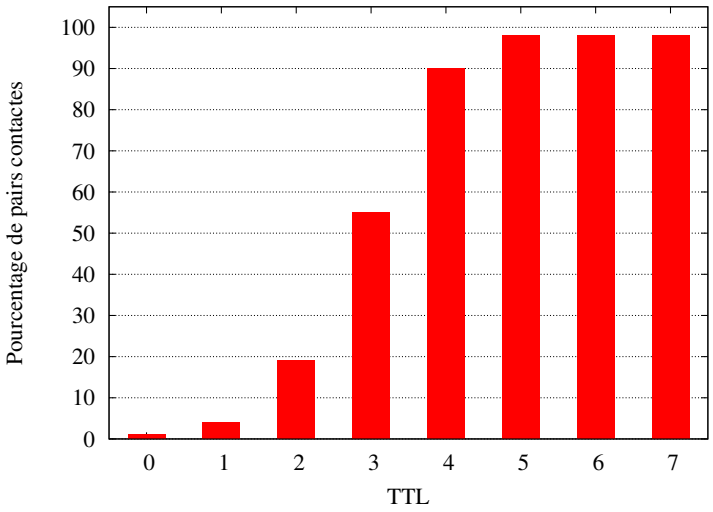

FIG. 4.9 – Pourcentage moyen de pairs atteints par les requêtes en fonction du TTL.

- il ne nécessite pas une connaissance du voisinnage pour effectuer le routage : l'algorithme est plus simple à mettre en place et l'exécution demande moins de ressources (temps, mémoire, etc.) ;
- il ne nous semble pas pertinent d'essayer de minimiser le nombre de messages (requêtes ou réponses) échangés car nous voulons mesurer les performances des méthodes de RI en terme de précision et de rappel.

Lors de l'émission d'une requête, l'algorithme Fully Distributed envoie la requête à tous les pairs se trouvant dans le diamètre TTL : il faut donc choisir une valeur pour le TTL. Pour pouvoir comparer les résultats obtenus en cadre distribué avec les résultats obtenus en cadre centralisé, il faut que tous les pairs accessibles soient atteints. Dans le cas contraire nous ne pourrons pas déterminer si la différence entre les résultats est dû au fait que :

- les documents ne sont pas tous accessibles,
- l'hétérogénéité est plus handicapante lorsque le nombre de pairs augmente.

Il faut choisir une valeur de TTL permettant d'accèder à tous les pairs. La figure 4.9 montre qu'un TTL de 8 est suffisant pour accéder à tous les pairs accessibles. On remarque qu'on n'accède jamais à 100% des pairs ; cela est dû au fait que certains pairs sont isolés, ou que certaines pairs quittent le système en cours d'exécution. Cette configuration semble réaliste étant données les caractéristiques des systèmes P2P.

Dans un système P2P hétérogène, le routage de requête soulève une nouvelle question : quelle requête transmettre de pairs en pairs ? Autrement dit, quand le pair P reçoit une requête, que doit-il renvoyé à ses voisins ? La réponse n'est pas triviale et nous avons plusieurs pistes de réponses :

1. P renvoit la requête qu'il a reçue, même s'il ne l'a pas parfaitement comprise. Avec un tel routage, la requête initiale parvient à tous les pairs.
2. P renvoit la requête telle qu'il a pû la comprendre. Cette stratégie dégrade la requête initiale au fur et à mesure qu'elle traverse le système.

3. P renvoit la requête qu'il a reçue et l'accompagne de la requête telle qu'il l'a comprise. Dans ce cas, la quantité d'information transférée augmente lors du parcours mais elle permet à ceux qui la reçoivent d'avoir différentes versions de la requête : l'originale et les versions intermédiaires.

La première solution nécessite que chaque pair ait une connaissance globale des alignements. Cette hypothèse ne nous a pas semblé réaliste d'autant plus que le nombre d'ontologies utilisées peut être important. La deuxième solution va davantage dégrader les résultats (car la requête est dégradée au cours du routage) mais elle est plus simple à mettre en œuvre que la troisième. Bien sûr, il sera intéressant de mettre en place les autres stratégies.

Interprétation des résultats

La figure 4.10 présente les résultats que nous avons obtenus dans un cadre distribué. Pour commencer, nous pouvons remarquer que le cosinus obtient de moins bons résultats que dans un cadre centralisé. Cela est logique puisques les requêtes sont appauvries entre chaques couples de pairs. Nous notons également qu'il obtient de relativement bons résultats : la courbe n'est pas beaucoup en dessous de la droite $f(x) = x$ (la courbe de précision est même parfois au dessus). Intuitivement, nous pouvons imaginer que la courbe serait plus basse si le système contenait plus de pairs. Cela reste évidemment à confirmer avec de nouvelles expérimentations.

Par ailleurs, notons que la courbe correspondant à la méthode ExSI^{2}D ne contient que trois points. Cela s'explique par le fait que les expérimentations sont longues à exécuter (entre 15 et 20h pour chaque point). De plus certains problèmes de programmation nous obligent parfois à recommencer une série de simulations. Ces conditions ne nous permettent pas d'interpréter correctement les résultats mais nous pouvons tout de même remarquer que :

- les points correspondant à 0 et 100% d'interopérabilité obtiennent de meilleurs résultats que le cosinus ;
- le rappel obtenu avec la méthode ExSI^{2}D est légèrement meilleur que celui du cosinus : cela semble logique et cohérent par rapport au cadre centralisé ;
- la précision obtenue avec la méthode ExSI^{2}D est moins bonne que celle du cosinus : cela est inattendu mais nous pouvons penser qu'en considérant tous les degré d'interopérabilité, la courbe de la méthode ExSI^{2}D sera globalement supérieure à celle du cosinus.

Les résultats obtenus sont mitigés et appelent évidemment à de nouvelles mesures.

4.3 Conclusion / Discussion

Les expérimentations ont permi de confirmer les résultats obtenus par Anthony VENTREQUE dans [34] : c'est ce que nous avons présenté dans la section 4.2.3. Par ailleurs, nous avons comparé la méthode ExSI^{2}D à la méthode basique utilisant le cosinus dans un cadre distribué (section 4.2.4).

Même si nous avons choisi les paramètres d'expérimentation de manière à être le plus proche possible de cas réels, il faut reconnaître qu'ils ne sont pas toujours réalistes. Par exemple la gestion de l'interopérabilité, et en particulier l'utilisation d'une seule ontologie, favorise la méthode ExSI^{2}D car elle permet théoriquement de toujours retrouver les concepts non-partagés. Par ailleurs la répartition des documents sur les pairs fait qu'un document pertinent pour une requête peut être inaccessible sémantiquement

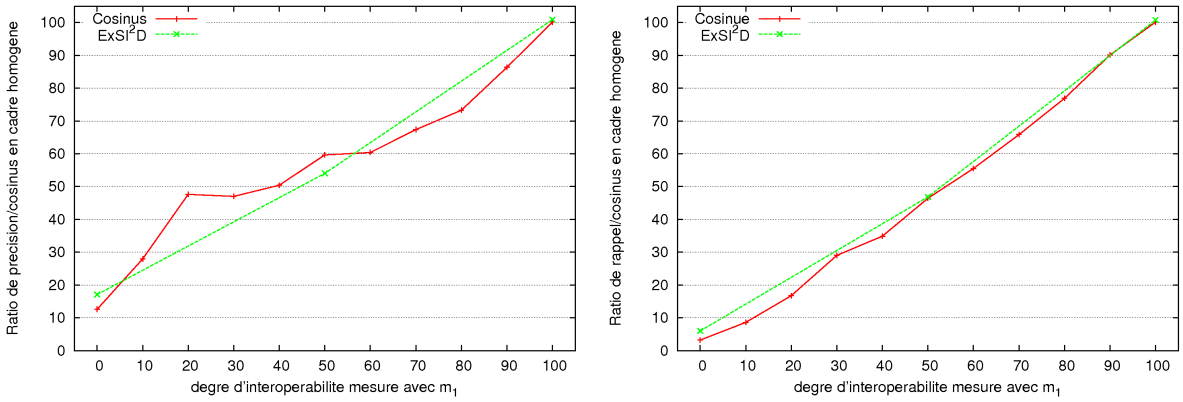

FIG. 4.10 – Évolution, dans un système distribué, du ration de précision (à gauche) et de rappel (à droite) entre les deux méthodes (cosinus et ExSI^{2}D) et le cosinus dans le cadre homogène, en fonction du degré d'interopérabilité..

car il se trouve trop loin de l'emetteur de la requête. Par « inacessible sémantiquement », nous voulons dire que la distance entre les pairs (l'emetteur de la requête et le pair contenant le document pertinent) et le degré d'interopérabilité font que la requête parvient au fournisseur d'information déformée : la requête n'est plus la même et il se peut que le document ne soit plus pertinent pour la requête. Si l'information contenue dans la requête n'est pas suffisante (s'il n'y a plus de concept dans le vecteur sémantique), la requête n'atteindra même pas le pair qui contient le document pertinent. Pour contrer cet effet il y a plusieurs solutions.

Premièrement, nous pouvons changer la manière de répartir les ontologies. En effet si nous faisons en sorte de grouper les pairs en fonction de l'ontologie qu'ils utilisent (pour créer des zones où la compréhension est bonne), les résultats des systèmes de RI seront meilleurs. Pour pouvoir regrouper les pairs de cette façon, il faut disposer de mesures permettant de quantifier le degré d'interopérabilité entre ontologies : les mesures proposées dans le chapitre 3 peuvent être utilisées.

Deuxièmement, nous pouvons faire varier le nombre d'ontologies utilisées par le système. Dans les expérimentations, nous avons choisi d'utiliser 10 ontologies différentes ; cela fait qu'en moyenne un ontologie est utilisée par 140 pairs. Il est difficile de dire si ce nombre est réaliste et nous pensons qu'il serait intéressant de le faire varier.

Troisièmement, nous pouvons faire varier le degré de réplication des données dans le système. Dans les expérimentations nous avons stocké dans un pair l'ensemble des documents écrits par un (ou plusieurs) auteur(s) ; donc seuls les documents écrits par plusieurs auteurs sont répliqués. Une autre stratégie serait de stocker dans les pairs l'ensemble des documents écrits par un auteur donné ainsi que les documents écrits par les auteurs ayant écrits un article avec l'auteur en question.

Enfin nous pouvons changer de stratégie pour la gestion de l'interopérabilité (cf. section 4.2.4, §4), et les mesures m_2 et m_3 pourraient également être utilisées pour mesurer l'interopérabilité (cf. section 3).

Chapitre 5

Conclusion et perspectives

L'objectif du travail présenté dans ce rapport était de mettre en place la méthode ExSI^{2}D dans un système P2P hétérogène. Cet objectif supposait de travailler sur trois axes : la recherche de mesures pour quantifier le degré d'interopérabilité entre les ontologies utilisées dans le système ; de la modélisation et du développement pour la mise en place de la méthode ExSI^{2}D et améliorer l'outils PeerSim ; et enfin l'expérimentation de la méthode dans un système P2P simulé avec PeerSim.

Le premier axe a abouti à la définition de trois mesures d'interopérabilité sémantique. Ces dernières utilisent des alignement d'ontologies pour définir à quel point deux ontologies sont compables de se comprendre, d'interopérer. La première mesure est basée sur le nombre de concepts partagés et non partagés entre deux ontologies. La deuxième utilise la notion de densité autour d'un concept. Enfin la troisième mesure prendre en compte le désordre des concepts entre les deux ontologies.

Le deuxième axe de travail concerne la modélisation de vecteurs sémantiques dans l'optique de les utiliser dans un cadre distribué et la modélisation de l'architecture orientée service de la méthode ExSI^{2}D. Le travail réalisé concerne également le refactoring de l'outils de simulation PeerSim pour le rendre générique (et donc réutilisable).

Enfin le troisième axe de travail regroupe l'expérimentation de la méthode ExSI^{2}D pour la comparer à la méthode basique (utilisant le cosinus pour calculer la pertinence d'un document par rapport à une requête). Avec ces expérimentations nous avons aussi pû comparer la qualité des deux méthodes de RI dans un cadre distribué et hétérogène et observer leurs comportements face à l'évolution du degré d'interopérabilité (le degré étant mesuré avec l'une des mesures proposées dans la section 3).

Ces trois axes de travail nous ont amené à proposer des solutions à des problèmes intermédiaires : placement des données, gestion de l'hétérogénéité dans le système P2P, etc. Un certain nombre de problèmes sont encore à adresser et cela ouvre des perspectives de travail (cf. section 4.3).

À court terme nous souhaitons mener à bien les expérimentations concernant la méthode ExSI^{2}D dans le cadre distribué en faisant varier différents paramètres : réplication des documents, taille du système, etc. Nous souhaitons aussi implémenter les différentes mesures d'interopérabilité sémantique pour pouvoir les utiliser lors du routage des informations dans le système. Cela devrait avoir pour effet d'améliorer la qualité des méthodes de RI et d'en améliorer les performances.

Bibliographie

[1] The chord project. http ://pdos.csail.mit.edu/chord/.

[2] Edutella - p2p for the semantic web. http ://www.edutella.org/edutella.shtml.

[3] The free haven project. http ://www.freehaven.net/.

[4] Grid5000. http ://www.grid5000.fr/.

[5] Jdom. http ://www.jdom.org.

[6] Jwnl. http ://jwordnet.sourceforge.net/.

[7] Lucene. http ://lucene.apache.org.

[8] The p-grid project. http ://www.p-grid.org/.

[9] Pastry. http ://research.microsoft.com/en-us/um/people/antr/Pastry/.

[10] Bernard Traversat Ahkil, Bernard Traversat, Ahkil Arora, Mohamed Abdelaziz, Mike Duigou, Carl Haywood, Jean christophe Hugly, Eric Pouyoul, and Bill Yeager. Project jxta 2.0 super-peer virtual network. Technical report, 2003.

[11] Reza Akbarinia, Esther Pacitti, and Patrick Valduriez. Reducing network traffic in unstructured p2p systems using top-k queries. *Distrib. Parallel Databases*, 19(2-3) :67–86, 2006.

[12] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999.

[13] Romaric Besançon, Mohand Boughanem, Chaudiron Stéphane, Jean-Pierre Chevallet, Eric Crestan, Hubert Fondin, Majid Ihadjadene, Wessel Kraaij, Claude De Loupy, Jian-Yun Nie, Jacques Savoy, and Lynda Tamine. *Les systèmes de recherche d'informations*. Lavoisier, 2004. § 8.2.1.

[14] Alain Bidault. *Affinement de requêtes dans un médiateur*. PhD thesis, 2002.

[15] Mohand Boughanem and Jacques Savoy, editors. *Recherche d'information états des lieux et perspectives*. Hermès Science Publications, 2008.

[16] Paolo Bouquet, Marc Ehrig, Jerome Euzenat, Enrico Franconi, Pascal Hitzler, Markus Krötzsch, Luciano Serafini, Giorgos Stamou, York Sure, and Sergio Tessaris. Specification of a common framework for characterizing alignment. Knowledge Web Deliverable 2.2.1v2, University of Karlsruhe, December 2004.

[17] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Computer Networks and ISDN Systems*, pages 107–117, 1998.

[18] Jérôme David and Jérôme Euzenat. Comparison between ontology distances (preliminary results). In *ISWC '08 : Proceedings of the 7th International Conference on The Semantic Web*, pages 245–260, Berlin, Heidelberg, 2008. Springer-Verlag.

[19] Gregor Heinrich. Free text corpora and their application to community retrieval evaluation. Technical report, Abrylon & University of Leipzig, April 2007.

[20] F. Holz, H. F. Witschel, G. Heinrich, G. Heyer, and S. Teresniak. An Evaluation Framework for Semantic Search in P2P Networks. In *Proceedings of the I2CS 2007*, 2007.

[21] Márk Jelasity, Alberto Montresor, Gian Paolo Jesi, and Spyros Voulgaris. The Peersim simulator. `http ://peersim.sf.net`.

[22] J. J. Jiang and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *International Conference Research on Computational Linguistics (ROCLING X)*, September 1997.

[23] M. Jovanovic, F. Annexstein, and K. Berman. Scalability issues in large peer-to-peer networks - a case study of gnutella, 2001.

[24] Alignment Coordinator Jrme, Coordinator Jérôme Euzenat (inria, Thanh Le Bach (inria, Jesús Barrasa (up. Madrid, Paolo Bouquet (u, Jan De Bo (vu Brussels, Rose Dieng (inria, Marc Ehrig (u, Ruben Lara (u. Innsbruck, Diana Maynard (sheffield U, Amedeo Napoli (cnrs/inria, Giorgos Stamou (ntu Athens, Pavel Shvaiko (u. Trento, Contact Person Dieter Fensel, and Contact Person Alain Leger. D2.2.3 : State of the art on ontology, 2004.

[25] Vana Kalogeraki, Dimitrios Gunopulos, and D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks, 2002.

[26] Niko Kotilainen, Mikko Vapa, Teemu Keltanen, Annemari Auvinen, and Jarkko Vuori. P2prealm – peer-to-peer network simulator, in. In *Proc. 11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks, 2006*, pages 93–99. Unpublished, 2006.

[27] Damien Levin, Louis-Alexandre Celton, and Gaëtan Hervouet. Une architecture de web services pour la recherche d'informations sémantique. Ter - master 1, Université de Nantes, 2009.

[28] Thomas Neumann, Matthias Bender, Sebastian Michel, and Gerhard Weikum. A reproducible benchmark for p2p retrieval. In *ExpDB*, pages 1–8, 2006.

[29] Bellot Patrice, Mohand Boughanem, Bruno Emmanuel, Favier Laurence, Fluhr Christian, Folch Helka, Grau Brigitte, Habert Benoît, Hascoët Mountaz, Le Maitre Ihadjadene, Majid Jacques, Murisasco Elisabeth, Papy Fabrice, Saleh Imad, Jacques Savoy, Tmar Mohamed, and Tebri Hamid. *Méthodes avancées pour les systèmes de recherche d'informations*. Lavoisier, 2004. § 4.4.

[30] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *In Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, 1995.

[31] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the gnutella network : Properties of large-scale peer-to-peer systems and implications for system design. Sep 2002.

[32] S. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *In Text REtrieval Conference (TREC-3)*, pages 109–126, 1996.

[33] Cerqueus Thomas, Charlotte Gueye, Koïta Aboubakar, Letavernier Camille, Masson Jérémy, and Pelleau Véronique. Projet de fin d'études - PeerUnit, Mars 2009. Encadrement : Gerson Sunye et Eduardo Almeida.

[34] Anthony Ventresque. *Espaces vectoriels sémantiques : enrichissement et interprétation de requêtes dans un système d'information distribué et hétérogène*. PhD thesis, September 2008.

[35] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 133 –138, New Mexico State University, Las Cruces, New Mexico, 1994.

LABORATOIRE D'INFORMATIQUE DE NANTES ATLANTIQUE

Mise en place de la méthode ExSI^{2}D dans un système P2P hétérogène

Thomas CERQUEUS
(encadré par Sylvie CAZALENS et Patrick VALDURIEZ)

Résumé

L'objectif du travail présenté dans ce rapport est la mise en place de la méthode ExSI^{2}D dans un système P2P hétérogène.

Pour atteindre cet objectif, nous proposons des structures sémantiques adaptées au contexte distribué et hétérogène : vecteur sémantique, explication de concept, explication de requête, etc. La méthode ExSI^{2}D étant composée de plusieurs modules indépendants nous proposons naturellement une application orientée services.

Par ailleurs, la notion d'hétérogénéité nous pousse à définir des mesures d'interopérabilité sémantique entre ontologies. Ces mesures peuvent, entre autre, servir lors du routage d'informations dans les systèmes P2P hétérogènes.

Le travail a abouti à un certain nombre d'expérimentations visant à comparer la méthode ExSI^{2}D à la méthode basique utilisant le cosinus ; et à observer le comportement de ces méthodes face à la variation de l'interopérabilité sémantique entre les participants. Les expérimentations ont été menées dans un cadre centralisé (seulement deux participants) ; puis dans un système P2P simulé avec l'outils PeerSim.

LINA, Faculté des Sciences & Techniques de Nantes
2, rue de la Houssinière
B.P. 92208 — F-44322 NANTES CEDEX 3

UNIVERSITÉ DE NANTES