# Proximité entre Requêtes dans un Contexte Médiateur
# Queries Proximity in a Mediator Context

A. Bidault, C. Froidevaux et B. Safar
L.R.I., UMR 8623 C.N.R.S - Université Paris-Sud
Bâtiment 490, 91405, Orsay Cedex, France
{bidault, chris, safar}@lri.fr

## Résumé

*La recherche d'informations au sein de sources multiples et hétérogènes est complexe. Les approches médiateur proposent des solutions qui exploitent une ontologie du domaine. Dans ce papier, nous décrivons comment l'ontologie peut aider à évaluer la proximité de deux requêtes et permet, lorsque la requête posée par l'utilisateur est en échec, de choisir une requête proche. Le travail présenté a été effectué au sein du projet* PICSEL[1]*. Les exemples cités sont issus du domaine des produits du tourisme, domaine d'expérimentation choisi au sein de ce projet.*

## Mots Clef

Coopération personne-système, Représentation de connaissances, Similarité entre Concepts.

## Abstract

*Information Retrieval in multiple and heterogeneous sources is complex. Mediator Approaches propose solutions that exploit a domain ontology. In this paper, we describe how an ontology can help to evaluate the proximity of two queries and permits, when the user's query fails, to choose a close query. The work presented has been developped in the* PICSEL *project. Examples come from the domain of tourism, the experimentation domain chosen for this project.*

## 1 Introduction

Avec le développement des réseaux, il est possible d'accéder aujourd'hui à de nombreuses sources d'information hétérogènes, réparties et autonomes.
Divers travaux s'intéressent au problème de l'intégration de sources d'information, et des systèmes tels que Information Manifold [13], Sims [2] ou PICSEL [9] ont vu le jour. Les concepteurs de ces systèmes voient l'intégration d'information comme la construction de *médiateurs* entre des utilisateurs et différents services d'information relatifs à un même domaine d'application. Un médiateur est un système à base de connaissances qui regroupe, dans un cadre unificateur et homogène, des connaissances propres au domaine d'application et une description sémantique et abstraite du contenu des différentes sources d'information disponibles. Les données sont stockées au niveau des sources d'information sous une forme qui peut varier d'une source à l'autre. Le médiateur joue le rôle d'interface entre l'utilisateur et les sources en lui donnant l'impression qu'il interroge un système d'information centralisé et homogène. Il reformule les requêtes posées dans le vocabulaire du domaine, en un ensemble de requêtes spécialisées, exprimées dans le vocabulaire des différentes sources d'information disponibles.

Décider quelles sources exploiter et comment les combiner pour satisfaire une requête devient alors un problème de raisonnement sur des requêtes, sur des connaissances décrivant un domaine d'application (e.g. le tourisme), et sur des connaissances décrivant le contenu de plusieurs sources d'information.

Nous nous intéressons ici aux requêtes qui n'obtiennent pas de réponse. Ce cas arrive fréquemment lorsque l'utilisateur ne maîtrise pas le domaine d'application et pose une requête violant les connaissances du domaine, ou ne connaît pas les limites des compétences des différentes sources disponibles par le médiateur. Notre travail consiste à proposer à l'utilisateur une requête qui apporte des réponses et qui reste sémantiquement proche de sa requête initiale pour le satisfaire. La réparation des requêtes violant le domaine d'application a été étudiée dans [4] ; nous nous intéressons ici aux requêtes qui ne peuvent pas être reformulées, par le médiateur, dans le vocabulaire des sources disponibles.

Notre approche est fondée sur la logique qui présente l'intérêt de fournir une sémantique formelle claire et rigoureuse de la description du domaine et du contenu des sources (cf. section 2).

Nous nous appuyons sur une base de requêtes prédéfinies initialisée à partir des sources (cf. section 4), et évolutive. Sur ce point, notre solution est donc spécifique de l'approche médiateur adoptée. À partir d'une hiérarchie de concepts et d'une requête utilisateur ne pouvant être reformulée, nous définissons une méthode de calcul de proximité entre concepts (cf. section 5) qui

1. PICSEL est un projet développé dans le cadre d'un contrat CTI $n°97$ 1B 378, avec France Telecom R&D.

nous permet de déterminer si une requête prédéfinie est suffisamment proche de celle de l'utilisateur pour pouvoir la lui présenter comme *solution* (cf. section 6).

# 2 Présentation du Contexte

Précisons tout d'abord le cadre logique dans lequel nous nous situons, en décrivant le formalisme utilisé pour exprimer les connaissances prises en compte sur le domaine d'application (l'ontologie), le contenu des sources relatives à ce domaine, ainsi que les requêtes.

## 2.1 Connaissances du Domaine

L'expression des connaissances du domaine s'appuie sur une représentation déclarative de classes d'objets (Voyage, Séjour, Vol, Pays...) et de relations entre ces classes. Les connaissances du domaine s'expriment à partir d'atomes de la forme $p(\bar{X})$ où $p$ est un nom de prédicat et $\bar{X}$ un n-uplet de variables. Certains prédicats unaires, appelés *concepts*, sont distingués car ils représentent des classes d'objets pertinentes dans le domaine d'application. $C(x)$ est un atome-concept si $C$ est un concept.

Les connaissances sont réparties en deux ensembles :

- **Un ensemble de règles** $\mathcal{D}$, de la forme :

$$r \ : p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n) \Rightarrow q(\bar{Y})$$

où $p_i(\bar{X}_i), q(\bar{Y})$ sont des atomes, $\bar{Y} \subseteq \cup_{i \in [1..n]} \bar{X}_i$.

**Exemple :** Un vol part du pays de sa ville de départ.
Vol(x) ∧ VilleDép(x, y) ∧ VilleDe(p, y) ⇒ PaysD(x, p)

Parmi les règles, on distingue l'ensemble $\mathcal{D}_\mathbf{h}$ des règles qui traduisent une hiérarchie entre les concepts du domaine : $C_1(x) \Rightarrow C_2(x)$, où $C_1$ et $C_2$ sont des concepts, $C_1$ est un **fils** de $C_2$ et $C_2$ un **père** de $C_1$.

**Exemple :** Un vol direct est un vol.
VolDirect(X) ⇒ Vol(X)

Les concepts et les prédicats n'apparaissant pas en conclusion d'une règle sont des **prédicats de base**. Les autres prédicats sont dits **ordinaires**. De plus, tous les prédicats qui ne sont pas des concepts sont dits **prédicat-nc**.

- **Un ensemble de contraintes** $\mathcal{C}$, de la forme

$$c \ : l_1(\bar{X}_1) \wedge \ldots \wedge l_n(\bar{X}_n) \Rightarrow \bot$$

avec $l_i(\bar{X}_i)(i \in [1..n])$ des **littéraux**, dont au plus un littéral négatif. Ces contraintes permettent d'exprimer des exclusions entre littéraux du domaine.

**Exemple :** Un voyage n'est pas un séjour.
Voyage(x) ∧ Séjour(x) ⇒ ⊥

Parmi les contraintes, on distingue l'ensemble des **contraintes de typage** $\mathcal{C}_t$ qui précisent pour un prédicat-nc, le concept qui **type** un de ses arguments.

$$p(\bar{X}) \wedge \neg C(x_i) \Rightarrow \bot \text{ avec } x_i \in \bar{X}.$$

Elles permettent de préciser à quel concept $C$ est lié chaque argument de $p(\bar{X})$.

**Exemple :** Le prédicat VilleDépart a des arguments de type voyage et lieu géographique.

VilleDépart(x,y) ∧ ¬ Voyage(x) ⇒ ⊥
VilleDépart(x,y) ∧ ¬ LieuGéographique(y) ⇒ ⊥

**La sémantique** de ces différents ensembles est la sémantique standard de la logique du premier ordre. Une interprétation $\mathcal{I}$ est définie sur un domaine non vide $\mathcal{O}^\mathcal{I}$. À chaque prédicat d'arité $n$, on assigne une relation d'arité n sur le domaine $\mathcal{O}^\mathcal{I}$.
- $\mathcal{I}$ est un modèle d'une règle $r$ si pour chaque assignation $\alpha$ des variables de la règle $r$ à des éléments du domaine $\mathcal{O}^\mathcal{I}$, avec $\alpha(\bar{X}_i) \in {p_i}^\mathcal{I}$ pour chaque $p_i$, on a $\alpha(\bar{Y}) \in q^\mathcal{I}$.
- $\mathcal{I}$ est un modèle d'une contrainte $c$ si pour chaque assignation $\alpha$ des variables de $c$ avec des éléments du domaine $\mathcal{O}^\mathcal{I}$, on n'a pas simultanément pour tout $l_i$: $\alpha(\bar{X}_i) \in {l_i}^\mathcal{I}$.
- $\mathcal{I}$ est un modèle de $\mathcal{D}$ et $\mathcal{C}$ s'il est un modèle de chacun de ses éléments.

## 2.2 Description des Sources

Le contenu de chaque source $\mathcal{S}$ est représenté à partir d'un vocabulaire $\mathcal{V}_S$ constitué d'autant de noms de relations locales $v_i$ qu'il y a de prédicats de base du domaine dont on sait que la source $\mathcal{S}$ fournit des instances. Ces relations locales $v_i$ sont appelées des **vues**. La description du contenu d'une source $\mathcal{S}$ en termes de ses vues est constituée de :

- **Un ensemble $\mathcal{D}_v \subseteq \mathcal{D}$ d'implications logiques,** reliant chaque vue à la relation du domaine dont elle peut fournir des instances, $v_i(\bar{X}) \Rightarrow p(\bar{X})$.

**Exemple :** Les implications suivantes expriment le fait que la source $\mathcal{S}_1$ fournit des instances du concept *Hôtel* ainsi que des instances de la relation binaire *localisé*.
$\mathcal{V}_{S_1}$ : $V_1(x) \Rightarrow$ hôtel(x) $V_3(x, y) \Rightarrow$ localisé(x, y)

- **Un ensemble $\mathcal{C}_v \subseteq \mathcal{C}$ de contraintes** caractérisant les instances des vues, dont les littéraux $l_i, i \in [1..n]$ sont formés à partir de prédicats de base et/ou de vues.

**Exemple :** Les localisations issues de $\mathcal{S}_1$ ne concernent que les hôtels présents dans $\mathcal{S}_1$ : $V_3(x, y) \wedge \neg V_1(x) \Rightarrow \bot$

et sont en France : $V_1(x) \wedge Localis\acute{e}(x,y) \wedge \neg France(y) \Rightarrow \bot$

### 2.3 Requêtes, Réécritures de Requêtes

Les requêtes que nous considérons sont des conjonctions d'atomes de la forme :

$$Q(\bar{X}) : p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n)$$

où les $p_i$ sont des prédicats du domaine, autres que des vues, apparaissant dans $\mathcal{D} \cup \mathcal{C}$, et $\bar{X} = \bar{X}_1 \cup \ldots \cup \bar{X}_n$ sont les variables dont l'utilisateur veut obtenir des instances quand il pose sa requête.

Classiquement, poser une requête $Q(\bar{X})$ se définit relativement à une base de données $\mathcal{BD}$, éventuellement associée à une théorie du domaine $\mathcal{T}$, et consiste à déterminer si : $\mathcal{BD}, \mathcal{T} \models \exists \bar{X}, [p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n)]$

Dans le cas d'une réponse positive à cette question logique, la *réponse* à la requête est le k-uplet $\bar{b} = \bar{b}_1 \cup \ldots \cup \bar{b}_n$ tel que :

$$\mathcal{BD}, \mathcal{T} \models p_1(\bar{b}_1) \wedge \ldots \wedge p_n(\bar{b}_n)$$

Or, nous ne sommes pas dans la situation classique : dans une approche de type médiateur, nous n'avons pas accès aux contenus des sources de données mais à une abstraction de celles-ci, exprimée sous la forme d'un ensemble $\mathcal{V}$ de vues. Poser une requête à un médiateur consiste alors à rechercher les différentes réécritures de cette requête en termes de vues. Une réécriture d'une requête $Q(\bar{X})$ est une requête qui, avec la théorie du domaine, implique logiquement $Q(\bar{X})$.

**Définition 2.1 :** *Soit $Q(\bar{X})$ une requête. Soit $\mathcal{T}$ la description du domaine comprenant $\mathcal{D}_v$ et $\mathcal{C}_v$ la description des sources de ce domaine. Soit $Q_{\mathcal{R}}$ une requête conjonctive.*
- $Q_{\mathcal{R}}(\bar{X})$ *est une* **réécriture** *de* $Q(\bar{X})$ *ssi* $Q_{\mathcal{R}}(\bar{X}), \mathcal{T}$ *est satisfiable, et* $Q_{\mathcal{R}}(\bar{X}), \mathcal{T} \models Q(\bar{X})$
- $Q_{\mathcal{R}}(\bar{X})$ *est une* **réécriture terminale** *de* $Q(\bar{X})$ *ssi* $Q_{\mathcal{R}}(\bar{X})$ *est une réécriture de* $Q(\bar{X})$ *qui ne contient que des atomes-vues.*

L'ensemble des réécritures terminales d'une requête fournit les plans de requêtes spécialisés dont l'exécution permet d'obtenir les réponses à la requête initiale.

## 3 Position du Problème

Dans le contexte du médiateur PICSEL [9], où nous nous situons, un algorithme de calcul des réécritures terminales a été conçu et mis en œuvre en Java. Ce calcul procède par expansions successives de la requête initiale. Un pas d'expansion consiste en un pas de chaînage-arrière sur les règles décrivant le domaine ($\mathcal{D}$) y compris sur les implications $\mathcal{D}_v$ liant les vues et les relations du domaine. Chaque pas d'expansion fournit des réécritures potentielles de la requête initiale. Pour chaque réécriture potentielle obtenue à chaque pas d'expansion, on teste sa compatibilité avec les contraintes $\mathcal{C}$. À la fin de l'algorithme, les réécritures calculées sont terminales ou n'ont pas d'expansion possible.

**Définition 3.1 :** *Une* **Requête-Succès,** *pour une théorie du domaine $\mathcal{T}$, est une requête qui a au moins une réécriture terminale satisfiable.*

Nous intervenons quand le médiateur détermine que toutes les réécritures potentielles calculées sont insatisfiables avec les contraintes du domaine. À cet instant, sans avoir accès aux sources de données, on est sûr que la requête posée par l'utilisateur n'a pas de réponses pour les sources disponibles.

**Définition 3.2 :** *Une Requête est une* **Requête-Échec** *pour une théorie du domaine $\mathcal{T}$, si et seulement si elle n'est pas une Requête-Succès pour $\mathcal{T}$.*
*Une* **Solution** *pour une requête-échec Q est une requête-succès proche*[2] *de Q*

Dans cet article, nous nous limitons à l'étude des requête-échecs satisfiables par rapport au domaine, c-à-d, pour lesquelles le médiateur ne peut pas utiliser ses sources pour les réécrire entièrement. Les solutions à apporter à des requêtes non satisfiables ont été étudiées dans [4] et sont à rapprocher des travaux développés, dans le cadre des Bases de Données Déductives, par Motro [14] et les chercheurs de l'équipe de Minker [7], [8] et [10].

**Définition 3.3:** *Une* **Requête NR** *sur une théorie du domaine $\mathcal{T}$ est une Requête-Échec satisfiable sur $\mathcal{T}$.*

Godfrey [10] a montré que le calcul de toutes les solutions d'une requête est un problème NP-difficile. Au lieu d'effectuer une recherche complète mais coûteuse, nous préférons une méthode approchée mais plus rapide qui s'appuie sur une base de requête-succès prédéfinies. Nous présentons alors à l'utilisateur, parmi ces requêtes, celles que nous estimons assez proches de sa propre requête.
Dans cet article, on montre tout d'abord comment créer une base de requêtes à partir des descriptions des sources ; puis on explique comment évaluer si deux prédicats sont sémantiquement proches et si deux requêtes sont suffisamment proches pour que l'une puisse être proposée comme solution pour l'autre.

## 4 Initialisation de la Base

Les solutions à proposer à l'utilisateur sont issues d'une base de requête-succès prédéfinies, $\mathcal{B}_{rp}$, initialisée à partir des descriptions des sources. Une requête-succès est construite à partir d'atomes dont une même source peut fournir différentes instances : on s'assure ainsi

2. La notion de proximité fait l'objet de la section 6.

d'avoir au moins cette source pour réécrire complètement la requête.
Grâce à l'approche médiateur, nous pouvons réaliser une telle initialisation de la base en exploitant largement les descriptions des sources.

Nous ne présentons ici que la phase d'initialisation de la base de requêtes prédéfinies. Son enrichissement et sa mise à jour à partir des requêtes validées par les utilisateurs et du nombre de réponses réellement constatées en interrogeant effectivement les sources avec les requêtes prédéfinies, ne seront pas abordés ici.
De plus, le fait que les données des sources évoluent rapidement avec le temps n'est pas pris en compte ; notre approche est statique. Pour pallier l'absence éventuelle et ponctuelle de réponses d'une source, on crée des requêtes qui pourront être réécrites par plusieurs sources.

## 4.1 Nombre de Réécritures Potentielles d'un Prédicat

Pour favoriser la réécriture des requêtes prédéfinies dans de multiples sources, nous créons les requêtes à partir des prédicats ayant un nombre de réécritures potentielles maximal et que nous appellerons par la suite **prédicats les plus usités**.

Un prédicat peut être réécrit directement par une vue, et apparaître dans plusieurs sources.

**Définition 4.1:** *Soit p un prédicat et $S_i$ une source, le* **poids de** *p* **pour** *$S_i$, $poids(p, S_i)$, est le nombre de règles de la source concluant sur p.*
*Si le médiateur peut accéder à n sources, le* **score direct** *de p est la somme des poids de p pour chaque source $S_i$, $score_d(p) = \sum_{i=1}^{n} poids(p, S_i)$.*

**Exemple :** Soit la description de la source $S_1$ qui propose des hôtels en France et des hôtels de luxe :

$\mathbf{r_1}$ : $V_1(x) \Rightarrow hôtel(x)$ $\quad$ $\mathbf{r_2}$ : $V_2(x) \Rightarrow France(x)$
$\mathbf{r_3}$ : $V_3(x) \Rightarrow localisé(x, y)$ $\quad$ $\mathbf{r_4}$ : $V_4(x) \Rightarrow hôtel(x)$
$\mathbf{c_1}$ : $V_3(x, y) \wedge \neg V_1(x) \Rightarrow \perp$
$\mathbf{c_2}$ : $V_3(x, y) \wedge \neg V_2(y) \Rightarrow \perp$
$\mathbf{c_3}$ : $V_4(x) \wedge classe(x, y) \wedge \neg luxe(y) \Rightarrow \perp$

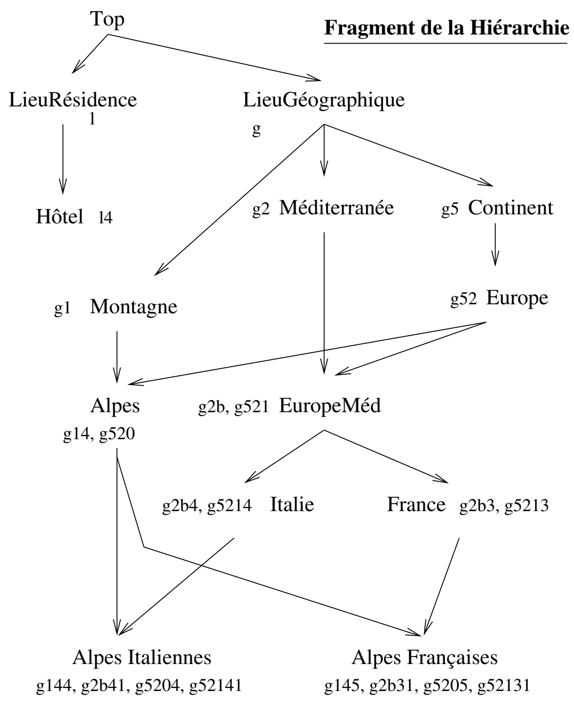
Si, hypothèse $H_1$, $S_1$ est l'unique source accessible par le médiateur, on obtient les scores directs suivants :
$score_d$(hôtel)= 2, $score_d$(France)= 1, $score_d$(LieuRésid.)= 0.

Le mécanisme d'expansion permet aussi de réécrire un concept en le remplaçant par un de ses fils. Nous définissons ci-dessous la descendance d'un prédicat.

**Définition 4.2 :** *L'***arbre de descendance** *d'un* **concept** *c, ad(c), est l'union de $\{c\}$ et des arbres de descendance de chacun de ses fils. L'arbre de descendance d'un* **prédicat-nc** *p est réduit au singleton $\{p\}$.*

**Exemple :** ad(LieuRésid.) = {LieuRésid., hôtel}
ad(France) = {France, AlpesFrançaises}

**Définition 4.3:** *Le* **score** *d'un prédicat p, score(p), est la somme des scores directs des prédicats de ad(p).*




**Exemple :** sous l'hypothèse $H_1$,
score(LieuRésid.) = $score_d$(LieuRésid.) + $score_d$(hôtel) = 2.
score(France) = $score_d$(France) + $score_d$(AlpesFrançaises) = 1.

**Définition 4.4:** *Un prédicat $p_1$ est* **plus usité** *qu'un prédicat $p_2$ si le score de $p_1$ est supérieur ou égal à celui de $p_2$.*

**Exemple :** sous l'hypothèse $H_1$, hôtel est plus usité que France.

## 4.2 Quota de Requêtes d'une Source

Après avoir défini sur quels prédicats créer les requêtes, on détermine maintenant le nombre de requêtes prédéfinies à créer pour une source donnée. Les sources disponibles sur le domaine (ex : *sncf, thalys, dégriftour*) étant de tailles très inégales dans leur contenu, le quota de requêtes à créer par source n'est pas fixe mais dépend du nombre de vues de chaque source.

Notons $T_B$ le nombre total de requêtes prédéfinies que doit contenir la base, ce nombre étant supposé fixé sur des critères d'efficacité de consultation de la base de requêtes par l'administrateur du médiateur.

**Définition 4.5:** *Soit une théorie du domaine $\mathcal{T}$, $T_B$ le nombre de requêtes prédéfinies à construire et M le nombre total de vues sur $\mathcal{T}$. Le* **quota global**, *$q_B$, de la théorie $\mathcal{T}$ est égal à $T_B/M$.*

**Exemple :** dans cet exemple et ceux qui suivent, on travaille avec l'hypothèse $H_2$ : il existe 3200 sources d'une taille moyenne de 5 vues.
Si $T_B$ vaut 4000, $\quad q_B = 4000/(3200 \times 5) = 0,25$.

**Définition 4.6:** *Soit $q_B$ le quota global d'une théorie du domaine $\mathcal{T}$ contenant une source S comportant $N_s$*

*vues. Le* **quota** $q_S$ *de* **la source** $S$ *est le nombre de requêtes qu'elle doit générer, et est égal à* $N_s \times q_B$.

**Exemple:** Sous $H_2$, le quota de $S_1$ est $q_{S1} = 4 \times 0,25 = 1$.

Le nombre $T_B$ étant fixé, la procédure de création des requêtes prédéfinies est la suivante

- Calculer les scores de chaque prédicat de la terminologie et le nombre M de vues sur $\mathcal{T}$.
- Pour chaque source S: (i) Calculer le quota $q_S$ en arrondissant à l'entier le plus proche.
  (ii) Construire les $q_S$ requêtes prédéfinies à partir des vues de S, cf. section 4.3.

Cette répartition en fonction du nombre de vues peut aboutir à ce que certaines sources, petites en nombre de vues, ne possèdent pas de requêtes prédéfinies à l'initialisation de la base ; si ces sources sont de bonne qualité, ce déséquilibre sera réparé pendant les étapes de mise à jour.

## 4.3 Construction des Requêtes Prédéfinies d'une Source

Les $q_S$ requêtes prédéfinies d'une source $S$ sont construites à partir de l'ensemble $P_S$ des prédicats dont la source fournit des instances. Le nombre de requêtes à construire étant limité, seuls les prédicats les plus usités interviendront dans les requêtes. Les éléments de $P_S$ sont donc ordonnés des plus aux moins usités. Un même prédicat pouvant être réécrit par plusieurs vues différentes d'une même source, le nombre de prédicats retenus pour créer les requêtes dépend du nombre de vues et est donné par la fonction définie ci-dessous.

**Définition 4.7:** *Soit une théorie du domaine* $\mathcal{T}$, *on appelle* **fonction de création** $f$ *de* $\mathcal{T}$ *la fonction qui associe au quota* $q_S$ *d'une source* $S$ *une approximation* $f(q_S)$ *du nombre de vues nécessaires pour générer* $q_S$ *requêtes à partir de* $S$.

Nos algorithmes fonctionnent quelle que soit la fonction $f$ choisie, mais celle-ci influe sur l'efficacité de la méthode. Si $f$ majore le nombre de vues à utiliser, de trop nombreuses requêtes seront générées et il faudra en supprimer ; si $f$ minore le nombre de vues, le nombre de requêtes sera insuffisant et devra être augmenté petit à petit. La fonction choisie ici pour l'exemple, $f(q_S) = 2q_S+1$, permet de retenir au moins trois vues distinctes, quand $q_S > 0$.
L'ensemble $P_{Su}$ des prédicats assez usités de la source $S$, est donc construit itérativement en choisissant un à un les prédicats, en commencant par le plus usité, jusqu'à ce que le nombre de vues permettant de réécrire tous les prédicats de $P_{Su}$ dépasse $f(q_S)$.

**Définition 4.8:** *Soit* $S$ *une source de quota* $q_S$, *et* $P_S$ *l'ensemble des prédicats dont elle fournit des instances. Soit* $P_{Su} \subseteq P_S$. $P_{Su}$ *est un ensemble des prédicats* **assez usités** *de* $S$, *si* $p_k$ *est l'un des prédicats les moins usités de* $P_{Su}$, *et :*
*- pour tout* $p_j$ *de* $P_S - P_{Su}$, $p_k$ *est plus usité que* $p_j$ *;*
- $\sum_{p_i \in P_{Su}-\{p_k\}} poids(p_i) < f(q_S) \leq \sum_{p_i \in P_{Su}} poids(p_i)$

**Exemple:** on suppose (hypothèse $H_3$ qui fixe arbitrairement l'ordre des plus usités et étend $H_2$), que sur l'ensemble des sources, *localisé* est plus usité que *hôtel* qui est plus usité que *France*.
Le nombre de vues à utiliser pour $S_1$ étant $f(q_{S1}) = 3$, on obtient: $P_{S_1u} = \{$localisé, hôtel$\}$.
En effet, poids(localisé, $S_1$) = 1, poids(hôtel, $S_1$) = 2, poids(France, $S_1$) = 1 ; si on ajoute France à $P_{S_1u}$, la somme des poids vaut 4, si on retire hôtel, la somme des poids vaut 1.

Les principales étapes de l'algorithme de construction des requêtes pour une source $S$ sont alors :

1. Déterminer l'ensemble $P_{Su}$ des prédicats assez usités de $S$.
   **Exemple:** $P_{S_1u} = \{$localisé, hôtel$\}$

2. Créer des requêtes ne contenant que des prédicats de $P_{Su}$ en étudiant les contraintes de $S$ pour déterminer les substitutions.
   **Exemple:** Les règles $r_1, r_3$ et $r_4$ font intervenir des prédicats de $P_{S_1u}$ et correspondent à l'ensemble de vues $\mathcal{E}_v = \{V_1, V_3, V_4\}$. $c_1$ ne prenant en compte que des vues de $\mathcal{E}_v$, elle permet de créer $Q_1$ = localisé(x, y) $\wedge$ hôtel(x).
   En revanche, la vue $V_4$ n'apparaît pas dans les contraintes avec d'autres vues de $\mathcal{E}_v$. Elle crée donc une conjonction ne contenant qu'un atome, la requête $Q_2$ = hôtel(x).

3. Saturer les contraintes de S pour spécialiser, vis-à-vis du domaine, les requêtes obtenues en 2., sans pour autant les contraindre vis-à-vis de $S$.
   **Exemple:** On regarde les contraintes qui font intervenir exactement un prédicat qui n'appartient pas au saturé courant de $Q_1$. La contrainte $c_2$ permet ainsi d'introduire le prédicat France dans $Q_1$.
   $Q_1$ = localisé(x, y) $\wedge$ hôtel(x) $\wedge$ France(y)
   La requête est ainsi plus spécialisée, mais peut toujours être réécrite dans $S_1$.
   Notons que $c_3$ ne peut pas être utilisée pour ajouter le prédicat *luxe* à $Q_2$ car le prédicat *classe* n'apparaît pas dans le saturé de $Q_2$.

4. Deux situations peuvent se présenter :
   - soit le nombre de requêtes créées est supérieur au quota de $S$ ; il faut alors supprimer une partie des requêtes créées ;
   - soit le quota n'est pas atteint. Il faut ajouter un élément à $P_{Su}$, et reprendre à l'étape 2.

**Remarques**: (i) Les requêtes avec un seul prédicat sont, *a priori*, trop générales pour être intéressantes, on décide donc de les supprimer.

**Exemple :** on supprime $Q_2$ et il ne reste plus que $Q_1$.

(ii) Les concepts les plus usités sont les plus généraux. Les requêtes générées à partir de ces concepts risquent d'être trop générales, et d'apporter plus de réponses que l'utilisateur ne peut en lire. Les étapes (2) et (3) permettent donc de les spécialiser : la première en faisant des liaisons de variables entre les prédicats les plus usités, la seconde en ajoutant des atomes.

Notre méthode de construction s'appuie sur un ensemble de prédicats assez usités ; notons cependant que lorsque le nombre de requêtes à créer par rapport au nombre de vues est assez élevé, il est moins coûteux de créer toutes les requêtes à partir de S puis d'enlever éventuellement celles qui seraient en trop.
Mais, un tel cas de figure sous-entend que le nombre de sources est faible, ou que le nombre de requêtes prédéfinies est élevé. Dans le premier cas, la base devient inutile puisqu'une approche complète et minimale peut être envisagée, dans le second, la recherche d'une requête proche de la requête de l'utilisateur peut être coûteuse, si on désire réellement utiliser la richesse de la base de requêtes.
Pour l'initialisation de cette base de requêtes, une maquette en java a été réalisée. Elle permet de créer un nombre plus ou moins grand de requêtes à partir d'un même ensemble de descriptions de sources.

### 4.4 Organisation des Requêtes

À partir d'une requête-échec $Q_u$ de l'utilisateur, on veut trouver rapidement dans notre base de requêtes prédéfinies $\mathcal{B}_{rp}$ des solutions pour $Q_u$. Nous présentons ici les méthodes d'indexation utilisées pour organiser les requêtes par clusters et éviter de parcourir toute la base.
Dans le cas général, un cluster $Cl$ est une subdivision homogène d'un ensemble de points $\mathcal{E}$, situés dans un espace $E$ contenant $\mathcal{E}$ ; soit $pt \in E$ le centre du cluster, alors $Cl$ regroupe tous les points de $\mathcal{E}$ dont la distance à $pt$ est minimale par rapport à la distance aux centres des autres clusters.

Nous utilisons la méthode des *k-medoids* [11] pour regrouper les requêtes relatives à une même thématique au sein de *clusters*, car elle procède par optimisation d'une subdivision existante et semble bien adaptée à une mise à jour de la base, contrairement à d'autres méthodes. Cette méthode est une adaptation, pour la logique du premier ordre, de la méthode des *k-means* [5][6], étudiée en logique propositionnelle. La méthode des k-means construit $k$ clusters en trois étapes : (i) déterminer $k$ centres pour les clusters, en les choissant dans $E$, (ii) regrouper les points de $\mathcal{E}$ autour de ces centres, en choisissant le centre le plus proche, (iii) redéterminer les nouveaux centres des clusters obtenus, pour minimiser les distances au sein d'un même cluster. On répète ces trois étapes jusqu'à ce qu'il y ait convergence.
Contrairement à la méthode des k-means, celle des k-medoids doit avoir un point de $\mathcal{E}$ comme centre ; en l'occurrence pour notre approche, il s'agit d'une requête prédéfinie de $\mathcal{B}_{rp}$, et non pas de n'importe quelle requête définissable (conjonction d'atomes).
Le regroupement et la minimisation des *distances*, dans notre cas, s'effectuent en utilisant la méthode de calcul décrite dans la section 6.

$Q_u$ sera alors confrontée aux requêtes du cluster de $\mathcal{B}_{rp}$ dont le centre est le plus proche de $Q_u$. Notons que ce centre ne sera pas forcément unique.

## 5 Proximité entre Prédicats

L'objectif de cette partie est de déterminer la proximité de différents concepts $C'$, $C''$, ..., regroupés au sein d'une hiérarchie, vis-à-vis d'un concept central $C$ de façon à pouvoir les ordonner des plus proches aux plus éloignés de $C$.

### 5.1 Travaux sur la Proximité

De nombreux travaux portent sur la proximité de concepts au sein d'une hiérarchie : l'approche la plus traditionnelle consiste à prendre le plus court chemin entre deux concepts $C_1$ et $C_2$ en passant par $C_g$, le concept le plus proche qui les généralise. Plus le chemin est court entre deux concepts, plus ils sont proches.

L'approche de Wu et Palmer [16] prend aussi en compte les méthodes de classification en calculant le palier maximal pour $C_1$ et $C_2$, c-à-d, le nombre d'arcs qui sépare le concept $C_g$ de la racine de la hiérarchie. Si $depth(C_g)$ est la valeur de ce palier et $depth_c(C_i)$ celui qui sépare $C_i$ de la racine en passant par $C_g$, la similarité de $C_1$ et $C_2$ est donnée par le calcul suivant :
$sim(C_1, C_2) = [2 \times depth(C_g)]/[depth_c(C_1) + depth_c(C_2)]$
Ce calcul, plus précis que le plus court chemin, ne permet pas de prendre en compte l'héritage multiple, c-à-d, l'existence pour deux concepts de plusieurs pères communs. La notion d'information partagée est, en revanche, à la base de la mesure de similarité de Lin [12] et de Resnik [15], mais ils introduisent pour ce faire des évaluations de fréquences sémantiques des concepts, recueillies sur corpus, informations dont nous ne disposons pas. De plus, ces travaux imposent de parcourir la hiérarchie à chaque comparaison, ce que nous souhaitons éviter.

Le calcul de la proximité de conjonction de prédicats est effectué par Andreasen [1] avec un langage assez proche du nôtre, qui mêle concepts et relations, dans le contexte de la recherche de documents. Dans ce contexte, chaque document est supposé indexé par un ensemble de descripteurs (ou concepts), $d = \{d_1, ..., d_m\}$. Etant donnée une mesure de similarité d'un concept à un descripteur, $sim(C, d_j), j \in [1..m]$, la similarité d'un concept à un document, $sim(C, d)$, est le *min* de la similarité du concept à chaque descripteur du document et si une requête $Q$ est constituée d'un ensemble de concepts $C_i$, sa similarité à un document est la simple moyenne de la somme des $sim(C_i, d)$. D'autre part, leur calcul de similarité ne tient pas encore compte des prédicats qui ne sont pas des concepts.

Nous souhaitons donc que notre notion de similarité utilise les connaissances exprimées au travers de la hiérarchie, et en particulier la notion d'héritage multiple, mais sans avoir à parcourir la hiérarchie à chaque calcul, et sans que nous ayons à introduire de nouvelles connaissances qualitatives pour chaque concept.

## 5.2 Principes de Base

Le calcul d'une note de proximité va, d'une part, s'appuyer sur la position des concepts dans la hiérarchie, et, d'autre part tenir compte du fait qu'elle doit servir à trouver une requête proche de celle de l'utilisateur.

Nous souhaitons ainsi que cette note vérifie les principes suivants :
- chaque concept de la hiérarchie recouvrant un certain nombre de caractéristiques, plus deux concepts partageront de caractéristiques, plus ils seront jugés proches.
- les spécialisations d'un concept étant jugées correctes vis-à-vis de ce concept et étant prises en compte dans le mécanisme de réécriture de requête du médiateur, elles doivent être jugées plus proches du concept que ses généralisants.

Pour mettre en œuvre ces principes, nous allons supposer que chaque arc de la hiérarchie traduit une caractéristique, et que l'ensemble des caractéristiques d'un concept se décompose en **caractéristiques définies** et **caractéristiques non définies** pour ce concept.

**Définition 5.1 :** *L'ensemble des* **caractéristiques définies** *d'un concept C est l'ensemble des occurrences d'arcs constituant les différents chemins* $Ch_1^\top$ *, ....,*$Ch_k^\top$ *menant de la racine Top* ($\top$) *de la hiérarchie à C.*

On note $P_h$, la **profondeur** de la hiérarchie.
Le nombre de caractéristiques d'un concept est alors $k \times P_h$, où $k$ est le nombre de chemins $Ch_i^\top$.
Cela revient à considérer que chaque chemin $Ch_i^\top$ a été prolongé par un chemin virtuel $Ch_i^\perp$ de $C$ vers $\perp$, de telle sorte que chaque chemin résultant de la concaténation de $Ch_i^\top$ et $Ch_i^\perp$ soit de longueur $P_h$.
L'ensemble des arcs des chemins virtuels représente, en quelque sorte, les **caractéristiques non définies** de $C$ et permet d'exprimer une étape de normalisation qui interviendra dans la suite du calcul.

**Exemple :** *EuropeMéd* possède deux chemins partant de $\top$, l'un passant par *Méditerranée*, $Ch_{Méd}^\top$, et l'autre par *Europe*, $Ch_{Eur}^\top$. $Ch_{Méd}^\top$ est de longueur 3, $Ch_{Eur}^\top$ de longueur 4, *EuropeMéd* a donc 7 caractéristiques définies.
Chaque chemin étant prolongé par un chemin virtuel pour que le chemin complet soit de longueur $P_h = 6$, la longueur de $Ch_{Méd}^\perp$ est donc de 3 et celle de $Ch_{Eur}^\perp$ de 2. *EuropeMéd* a donc $2 \times 6 = 12$ caractéristiques au total, 7 définies et 5 non définies.

Concrètement, la représentation des chemins menant de $\top$ à chaque concept $C$ est effectuée par une numérotation intrinsèque de tous les concepts de la hiérarchie qui traduit leur filiation.

## 5.3 Numérotation d'un Concept

**Définition 5.2:** *Soit une hiérarchie* $\mathcal{D}_h$*, un* **numéro** *pour un concept est une chaîne de caractères unique pour* $\mathcal{D}_h$*, définie sur un alphabet de n symboles. On appelle* **longueur** *le nombre de ses caractères.*

L'alphabet doit être suffisamment large pour prendre en compte le nombre maximal de fils que peut avoir un même concept dans $\mathcal{D}_h$. Dans nos exemples, n = 36 $(0, 1, .., 9, a, b, .., z)$ et cela semble suffisant.

Un concept a plusieurs numéros s'il possède plusieurs pères, ou si un de ses pères a plusieurs numéros.

**Définition 5.3: L'identifiant** *d'un concept C,* $id(C)$ *est l'ensemble de ses numéros.*

Tous les fils d'un concept héritent de l'identifiant de ce concept, et pour chaque fils, chaque numéro de l'identifiant hérité est étendu sur la droite par un chiffre distinct. La méthode utilisée pour effectuer cette numérotation est inspirée de celle du tri topologique, ce qui est possible puisque la hiérarchie est sans circuit et de racine *Top*. L'unique condition pour l'attribution des chiffres est que chaque numéro de la hiérarchie soit unique.

**Exemple :** les deux concepts France et Italie héritent de EuropeMéd [g521, g2b]. Ici, les numéros sont étendus par le chiffre 3 pour France [g5213, g2b3], et 4 pour Italie [g5214, g2b4].

A chaque numéro de l'identifiant d'un concept $C$ correspond un unique chemin de $\top$ vers ce concept. Si un numéro s'écrit $p_1...p_n$, il lui correspond un chemin de longueur $n$ tel que le $i^{ème}$ arc relie le concept ayant dans son identifiant le numéro $p_1...p_{i-1}$ au concept ayant dans son identifiant le numéro $p_1...p_{i-1}p_i$. La somme des longueurs des numéros de l'identifiant d'un concept est égale au nombre de caractéristiques définies du concept.

## 5.4 Calcul d'une Note

Nous définissons la **note de proximité** d'un concept $C'$ **centrée sur** un concept $C$, notée $N_{C' \to C}$, qui va permettre d'ordonner des concepts $C'$, des plus proches aux plus éloignés du concept central $C$. Les notes de proximité définies vont de 0 à 1 et leur calcul est basé sur :

(i) le décompte des caractéristiques communes aux généralisants communs de $C$ et $C'$, ce qui implique l'étude des différents numéros des identifiants de $C$ et $C'$,

(ii) le décompte des caractéristiques de $C'$ que $C$ ne possède pas.

Dans ce qui suit, nous allons utiliser les notations suivantes. L'identifiant du concept central $C$ est noté $id(C) = [n_1, ..., n_k]$, et celui de $C'$, $id(C') = [\ m_1, ..., m_l]$. Pour tout numéro $\mu$, on note $|\mu|$ sa longueur. + désigne le symbole de concaténation de deux chaînes de caractères. Etant donné deux numéros, $n_\alpha$ et $m_\beta$, où $\alpha \in [1..k]$ et $\beta \in [1..l]$, on note $com_{\alpha\beta}$ leur plus grand préfixe commun. On a donc les décompositions suivantes, $n_\alpha = com_{\alpha\beta} + fin_{n_\alpha}$ et $m_\beta = com_{\alpha\beta} + fin_{m_\beta}$.

Nous définissons tout d'abord le **rapport de proximité de $m_\beta$ centré sur $n_\alpha$**,
$R_{m_\beta \to n_\alpha} = ((|com_{\alpha\beta}| + P_h - |n_\alpha|)/P_h) - 0,002 \times |fin_{m_\beta}|$.

Ensuite, pour chaque numéro $n_\alpha$, on définit un rapport de proximité de $C'$ centré sur $n_\alpha$, $Rmax_{C' \to n_\alpha}$, qui mesure la proximité maximale des différents numéros de $C'$ centrée sur $n_\alpha$.

Le **rapport de proximité de $C'$ centré sur** $n_\alpha$ est :
$Rmax_{C' \to n_\alpha} = \text{MAX}\ \{R_{m_\beta \to n_\alpha},\ \beta \in [1..l]\}$.

La **note de proximité de $C'$ centrée sur** $C$ est :
$N_{C' \to C} = \text{MOY}\ \{Rmax_{C' \to n_\alpha},\ \alpha \in [1..k]\}$.

## 5.5 Exemples et Justifications

**Exemple** : calculons la note de proximité du concept $C' = Alpes$ centrée sur le concept $C = EuropeMéd$, avec $P_h = 6$.
id($Alpes$) = [g14, g520]   id($EuropeMéd$) = [g2b, g521]
$R_{g14 \to g2b} = (|g| + 6 - |g2b|)/6 - 0,002 \times |14|$
$= 4/6 - 0,004 = 0,663$
$R_{g520 \to g2b} = (|g| + 6 - |g2b|)/6 - 0,002 \times |520| = 0,661$
$R_{g14 \to g521} = (|g| + 6 - |g521|)/6 - 0,002 \times |14| = 0,496$
$R_{g520 \to g521} = (|g52| + 6 - |g521|)/6 - 0,002 \times |0| = 0,831$
$Rmax_{Alpes \to g2b} = 0,663$     $Rmax_{Alpes \to g521} = 0,831$
$N_{Alpes \to EuropeMéd}$ = (0,663 + 0,831)/2 = 0,747

Dans la définition du rapport de proximité d'un numéro centré sur un autre numéro, $R_{m_\beta \to n_\alpha}$, $|com_{\alpha\beta}|$ correspond au nombre de caractéristiques définies, communes aux deux numéros $n_\alpha$ et $m_\beta$. Le fait de conserver ensuite pour chaque $n_\alpha$, le maximum de ces rapports revient à s'intéresser au plus grand des plus grands préfixes communs des $m_\beta$ avec ce $n_\alpha$, c'est-à-dire, au plus petit commun généralisant (*ppcg*) à $C'$ et $C$, selon les caractéristiques exprimées dans $n_\alpha$.
En effet, pour deux concepts $C_1$ et $C_2$, les numéros correspondant aux identifiants des plus petits communs généralisants de $C_1$ et $C_2$ s'obtiennent en gardant l'ensemble des plus grands préfixes $|com_{12}|$ communs à $C_1$ et $C_2$. Il faut ensuite rechercher dans la table des concepts ceux qui sont identifiés par **au moins un** des $|com_{12}|$.

**Exemple**: Les plus grands des plus grands préfixes communs aux numéros de *Alpes* et *EuropMed* sont $\{g\}$ et $\{g52\}$, et correspondent aux numéros des concepts *LieuGéographique* et *Europe* qui sont leurs deux *ppcg*.

Le nombre $|com_{\alpha\beta}|$ est ensuite augmenté pour comptabiliser les caractéristiques *non définies* de $n_\alpha$, d'où le terme $P_h - |n_\alpha|$. En effet, les caractéristiques non définies de $n_\alpha$ sont communes à chacun de ses généralisants. La division par $P_h$ correspond à une phase de normalisation qui permet d'obtenir des rapports compris entre 0 et 1.
Cette première partie du calcul permet de reprendre l'intuition du point (i), c'est-à-dire la recherche des caractéristiques communes aux généralisants communs de $C$ et $C'$.

La prise en compte du point (ii), c'est-à-dire des caractéristiques de $C'$ que $C$ ne possède pas, s'effectue par l'introduction d'une pénalité qui fait intervenir la taille du suffixe de $m_\beta$, $|fin_{m_\beta}|$.

**Exemple**: *EuropeMéd* possède quatre descendants: *France* et *Italie* comme fils, *AlpesFrançaises* et *AlpesItaliennes* comme petit-fils, les petit-fils sont plus éloignés que les fils:
$N_{France \to EuropeMéd} = N_{Italie \to EuropeMéd}$
= ((6/6 - 0,002) + (6/6 - 0,002)) /2 = 0,998.
$N_{AlpesFr. \to EuropeMéd} = N_{AlpesIt. \to EuropeMéd}$
= ((6/6 - 0,004) + (6/6 - 0,004)) /2 = 0,996.

Plus généralement, $N_{C' \to C}$ = 0,998 si $C'$ est un fils de $C$, $N_{C' \to C}$ = 0,996 si $C'$ est un petit-fils de $C$.

## 5.6 Proximité entre Prédicat-nc

On s'intéresse maintenant aux prédicats qui ne sont pas des concepts. Le calcul de proximité s'appuie sur leur nombre d'arguments et sur les contraintes de typage qui précisent les concepts typant ces arguments ; notre choix est toujours de ne pas définir de connaissances sémantiques différentes de celles déjà présentes dans PICSEL.

**Définition 5.4:** *Soit $p_1(x_1, ..., x_n)$ et $p_2(y_1, ..., y_m)$ deux prédicat-nc. Soit $C'_j, j \in [1..m]$ les concepts qui*

*typent les arguments $y_j$, et $C_i, i \in [1..n]$ ceux qui typent les arguments $x_{i \in [1..n]}$. La* **note de proximité de** $p_2$ **centrée sur** $p_1$ *est :*
$N_{p_2 \to p_1} = \text{Moy}_{i \in [1..n]} \{ \text{Max}_{j \in [1..m]} \{N_{C'_j \to C_i}\} \}$.

Lorsqu'il n'existe pas de contrainte de typage pour un argument, on attribue la note zéro, c-à-d, la note de proximité du concept *Top* centrée sur n'importe quel concept.

**Exemple :** Soit les 5 contraintes de typage suivantes pour les prédicats $p_1$ = Équipement_Ass qui associe un équipement à un lieu de résidence, et $p_2$ = Dist_Équipement qui décrit le nombre d'équipements et la distance qui sépare ces équipements d'un lieu géographique :
Équipement_Ass(x,y) $\wedge \neg$ LieuRésidence(x) $\Rightarrow \perp$
Équipement_Ass(x,y) $\wedge \neg$ Équipement(y) $\Rightarrow \perp$
Dist_Équipement(x,y,z) $\wedge \neg$ LieuGéographique(x) $\Rightarrow \perp$
Dist_Équipement(x,y,z) $\wedge \neg$ Équipement(y) $\Rightarrow \perp$
Dist_Équipement(x,y,z) $\wedge \neg$ Nombre(z) $\Rightarrow \perp$

Et soit les notes de proximités suivantes (voir tableau), choisies pour cet exemple, où les $C_i$ sont les lignes du tableau, et les $C'_j$ les colonnes : $N_{LieuGéo. \to LieuRés.} = 0,4$

| Dist_Éq<br>Éqt_Ass | LieuGéo | Équipement | Nombre |
|---|---|---|---|
| LieuRésidence | **0,4** | 0,3 | 0 |
| Équipement | 0,1 | **1** | 0,2 |

À partir du tableau, où les n meilleures notes de proximité des $C'_j$ centrés sur les $C_i$ sont indiquées en gras, on calcule la note de proximité de Dist_Éq centrée sur Éqt_Ass qui vaut $(1 + 0,4)/2 = 0,7$ .

# 6 Proximité entre Requêtes

Nous présentons dans cette section comment comparer chacune des requêtes de $\mathcal{E}$ avec $Q_u$.
Le calcul des notes de proximité entre deux requêtes s'appuie sur les proximités entre les prédicats, mais doit également prendre en compte
(i) la différence des noms de variables de la requête utilisateur et des requêtes prédéfinies,
(ii) les liaisons entre ces variables.

**Substitutions** La proximité entre deux prédicats ne doit être calculée que si ces prédicats se réfèrent au même objet (même variable). Or, nous ne voulons pas calculer toutes les substitutions possibles entre les deux requêtes, puis les notes de proximité pour chacune de ces substitutions, et choisir celle qui offre la meilleure note, car cette méthode est trop coûteuse.
Une heuristique est donc utilisée pour choisir une substitution parmi les prétendantes.

**Algorithme :**
1. Comparer les prédicats de la requête prédéfinie $Q_p$ avec ceux de la requête utilisateur $Q_u$ ; la comparaison s'effectue dans ce sens car on souhaite déterminer quels sont les atomes de $Q_u$ les plus proches de ceux de $Q_p$.

**Exemple :** requête utilisateur
$Q_u = A(x) \wedge B(x,y) \wedge C(x,z) \wedge D(y,w) \wedge F(z)$,
requêtes prédéfinies
$Q_{p1} = A'(l) \wedge B'(l,k) \wedge D(m,n) \wedge F'(l)$ et
$Q_{p2} = A'(l) \wedge B'(k,l) \wedge D(m,n) \wedge F'(l)$
Avec les notes de proximité[3] suivantes ($N_{A \to A'} = 0,4$) :

| $Q_p \backslash Q_u$ | A | B | C | D | F |
|---|---|---|---|---|---|
| A' | **0,4** | 0 | 0 | 0,1 | 0,1 |
| B' | 0,1 | **0,6** | 0,1 | 0,2 | 0 |
| D | 0 | 0,2 | 0,3 | **1** | 0 |
| F' | 0,1 | 0 | 0,1 | 0 | **0,2** |

2. Établir un classement $\mathcal{C}$ des atomes de $Q_p$ dans l'ordre décroissant de leurs notes de proximité.

**Exemple :** Le classement pour $Q_{p1}$ et $Q_{p2}$ est : $[D, B', A', F']$

3. Ensuite, on applique les substitutions imposées par chaque couple de prédicats proches en commençant par le couple associé au premier prédicat de $\mathcal{C}$, jusqu'à ce que chacun des atomes de $Q_p$ soit instancié avec des variables présentes dans $Q_u$.

**Exemple :**

| | | | |
|---|---|---|---|
| $Q'_{p1}$ | D | $A'(l) \wedge B'(l,k) \wedge D(\mathbf{y},\mathbf{w}) \wedge F'(l)$ |
| | B' | $A'(\mathbf{x}) \wedge B'(\mathbf{x},\mathbf{y}) \wedge D(\mathbf{y},\mathbf{w}) \wedge F'(\mathbf{x})$ |
| | A'F' | le traitement est fini |
| $Q'_{p2}$ | D | $A'(l) \wedge B'(k,l) \wedge D(\mathbf{y},\mathbf{w}) \wedge F'(l)$ |
| | B' | $A'(\mathbf{y}) \wedge B'(\mathbf{x},\mathbf{y}) \wedge D(\mathbf{y},\mathbf{w}) \wedge F'(\mathbf{y})$ |
| | A'F' | le traitement est fini |

Maintenant qu'une requête prédéfinie a été totalement instanciée, on la note pour la comparer à celle de l'utilisateur.

**La note de $Q_p$ centrée sur $Q_u$** est la moyenne des meilleures notes obtenues par la comparaison avec chacun des atomes de $Q_u$ des atomes de $Q'_p$. Cette fois, les notes s'effectuent dans l'autre sens, puisque nous voulons déterminer quelles sont les requêtes $Q_p$ les plus proches de $Q_u$.

Une première note est obtenue en comparant les atomes utilisant les mêmes variables pour les mêmes objets. Pour contrebalancer le fait qu'une seule substitution est choisie, on calcule une deuxième note pour chaque atome de $Q_u$ et on prend la meilleure. Cette note est calculée uniquement sur les noms de prédicats, mais subit un malus qui peut consister, entre autres, à diviser la note par 2, *malus choisi pour notre exemple,* par 3, ou bien à retirer systématiquement $0,4$ ou $0,5$.

**Exemple :** avec ces notes de proximité ($N_{A' \to A} = 0,3$) :

| $Q_u \backslash Q_p$ | A' | B' | D' | F' |
|---|---|---|---|---|
| A | **0,3** | 0 | 0,1 | 0,1 |
| B | 0 | **0,7** | 0,1 | 0 |
| C | 0 | 0,1 | **0,2** | 0,1 |
| D | 0 | 0,2 | **1** | 0 |
| F | 0,1 | 0,1 | 0 | **0,4** |

3. $Q_{p1}$ et $Q_{p2}$ partageant les même prédicats, on ne présente qu'une seule table de proximités.

Les tableaux suivants montrent le calcul des deux notes pour $Q_{p1}$ et $Q_{p2}$ par rapport à $Q_u$.

| $Q_u$ | première note $Q'_{p1}$ | $Q'_{p2}$ | |
|---|---|---|---|
| A(x) | A'(x) **0,3** F'(x) 0,1 | **0** | |
| B(x, y) | B'(x, y) **0,7** | B'(x, y) **0,7** | |
| C(x, z) | **0** | **0** | |
| D(y, w) | D(y, w) **1** | D(y, w) **1** | |
| F(z) | **0** | **0** | |
| $Q_u$ | seconde note $Q_{p1}$ ou $Q_{p2}$ | $\max p_1$ | $\max p_2$ |
| A(x) | A'(x) **0,15** | 0,3 | 0,15 |
| B(x, y) | B'(x, y) **0,35** | 0,7 | 0,7 |
| C(x, z) | D(x, y) **0,1** | 0,1 | 0,1 |
| D(y, w) | D(y, w) **0,5** | 1 | 1 |
| F(z) | F'(w) **0,2** | 0,2 | 0,2 |
| | **Notes** | 0,46 | 0,43 |

On présentera à l'utilisateur les requêtes $Q_p$ de la base, celles avant substitution, en suivant l'ordre décroissant des notes obtenues par les requêtes après substitution : la requête $Q_{p1}$ sera donc présentée avant $Q_{p2}$.
Différents seuils sont définis par défaut ou par l'utilisateur ; ils correspondent par exemple aux notes minimales que doivent obtenir les requêtes prédéfinies pour être estimées proches de $Q_u$, ou au nombre de réponses à apporter à l'utilisateur.

# 7 Conclusion

L'objectif de ce travail est de proposer des solutions approchées à un utilisateur dont la requête initiale n'obtient pas de réponse. Ces solutions approchées sont extraites d'une base prédéfinie de requête-succès (i.e. qui peuvent se réécrire à partir des sources disponibles du médiateur). Nous avons défini un cadre formel permettant de caractériser les prédicats utilisables pour générer ces requête-succès, et des méthodes permettant d'évaluer la proximité de deux prédicats, puis de deux conjonctions de prédicats.
Ce travail a été implémenté en java.
De plus, plusieurs expérimentations sont effectuées pour déterminer (i) la taille souhaitable de la base de requêtes prédéfinies, (ii) celle de l'ensemble de confrontation, et (iii) différents seuils qui précisent si une requête (ou un concept) est proche d'une autre.
Concernant la mise à jour de la base de requêtes, nous envisageons des formalisations inspirées des techniques utilisées en raisonnement à base de cas.

# Références


[1] T. Andreasen, J.F. Nilsson and H.E. Thomsen. Ontology-Based Querying, In *Proc. of Fourth Int. Conf. on Flexible Query Answering Systems*, FQAS'2000, pp.15-26, Warsaw, Poland, October 2000.

[2] Y. Arens, C. A. Knoblock, W.-M. Shen. Query reformulation for dynamic information integration, Dans *J. Intelligent Information Systems*, Kluwer, 6(2), pp. 99-130, juin 1996.

[3] A. Bidault, C. Froidevaux, B. Safar. Repairing Queries in a Mediator Approach, Dans *Proc. of ECAI 2000,* pp. 406-410, 2000.

[4] G. Bisson. Conceptual Clustering in a First-Order Logic Representation, Dans *Proc. of AAAI conf.,* pp. 82-87, 1992.

[5] D. Cox. Note on Grouping, Dans *J. Am. Stat. Assoc.*, (52), pp. 543-547, 1957.

[6] E. Diday, J. Lemaire, J. Ponjet, F. Testu. Éléments d'Analyse des Données, Dunod, Paris, 1984.

[7] T. Gaasterland. Cooperative Answering through Controlled Query Relaxation, Dans *IEEE Expert*, pp. 48-59, sept-oct 1997.

[8] A. Gal. Construction de réponses coopératives à l'aide de contraintes d'intégrité dans des bases de données logiques, *Thèse de l'Université Rennes I*, déc. 1988.

[9] F. Goasdoué, V. Lattès, M.-C. Rousset. The Use of CARIN Language and Algorithms for Information Integration: The PICSEL Project, Dans *Int. J. of Cooperative Inf. Syst.* (IJCIS), 1999.

[10] P. Godfrey. Minimization in Cooperative Response to Failing Databases Queries, Dans *Int. J. of Cooperative Inf. Syst. (IJCIS)* 6(2), pp. 95-149, juin 1997.

[11] M. Kirsten, S. Wrobel. Extending k-means Clustering to 1st-Order Representations, Dans *Inductive Logic Programming*, LNAI 1866, pp. 112-126, 2000.

[12] D. Lin. An Information-Theoretic Definition of Similarity, In *Proc. of the Fifteenth Int. Conf. on Machine Learning (ICML-98*, pp. 296-304, Madison, Wisconsin, 1998.

[13] A. Y. Levy, A. Rajamaran, J. Ordille. Query-answering algorithms for information agents, Dans *Proc. of AAAI conf.*, pp. 40-47, août 1996.

[14] A. Motro. Intensional Answers to Database Queries, *IEEE*, 6(3), pp 444-454, juin 1994.

[15] P. Resnik. Semantic Similarity in a Taxonomy: An Information-based Measure and its Application to problem of Ambiguity in Natural Language, Dans *J. of Artificial Intelligence Research*, (11) pp. 95-130, 1999.

[16] Z. Wu and M. Palmer. Verb Semantics and Lexical Selection, Dans *Proc. of 32nd Annual Meeting of the Association for Computational Linguistics*, 1994.