Adaptation du modèle de langue pour le tri des réponses dans les BD

Abdelhamid Chellal, Mohand Boughanem, Karima Amrouche

Adaptation du modèle de langue pour le tri des réponses dans les BD

Abdelhamid CHELLAL *— Mohand BOUGHANEM *— Karima AMROUCHE **

**Institut de Recherche en Informatique de Toulouse,*
118 route de Narbonne 31000 Toulouse Cedex 9, France
{abdelhamid.chellal,bougha}@irit.fr
*** Ecole Supérieure Nationale de l'Informatique, Alger, Algérie*
K_amrouche@esi.dz

RÉSUMÉ. *L'information sur le web est de plus en plus extraite depuis des bases de données (BD) où les langages d'interrogation sont basés sur une recherche exacte. L'utilisateur se trouve confronté au problème de réponses nombreuses lorsque sa requête est peu sélective. Pour remédier à ce problème, plusieurs approches ont été proposées, à l'instar de celles utilisant les techniques de relaxation des requêtes. D'autres travaux proposent de classifier les résultats. Une autre classe d'approches, au quelle on s'intéresse, suggère l'adaptation des techniques de la recherche d'information (RI) pour trier les résultats dans les BD. On propose dans cet article, une adaptation du modèle de langue de la RI pour trier les tuples retournés selon leur score de pertinence vis-à-vis la requête. Ce score est évalué par un modèle de langue bi-gramme qui combine, à travers un lissage par interpolation, les probabilités d'occurrence des valeurs des attributs dans l'ensemble des tuples retournés ainsi que dans la BD. Nous avons évalué l'efficacité de notre approche sur une table contenant 16842 tuples. Les résultats préliminaires obtenus montrent l'intérêt d'exploiter les dépendances entre les valeurs d'attributs.*

. ABSTRACT. *Information over the web is increasingly retrieved from relational databases using exact query matching. As result, the user can be confronted to the problem of overabundant answers in the case of imprecise query. To overcome this problem, several approaches have been proposed we distinguish those based on automatic strengthening of the predicates used in vague conditions. Others works suggest the use of classifications techniques to provide users with hierarchical clustering schemas of their query results. Another class of approaches proposes the adaptation of information retrieval (IR) approaches for ranking tuples returned for a query. We discuss in this paper, an adaptation of IR based language model to rank results according to their relevance score. This score is evaluated using bi-gramm language model smoothed by interpolation between the probabilities of occurrence of attributes values in the result set and their probabilities of occurrence in the DB. We evaluate the effectiveness of our approach with a table with16842 tuples. The preliminary results show the interest of exploiting attribut's value dependency.*

MOTS-CLÉS : *Recherche d'information, tri dans les BD, problème réponses nombreuse, modèle de langue.*

KEYWORDS: *Information retrieval, DB rank, overabundant answers, language model.*

1. Introduction

Une multitude des bases de données telles que les bibliothèques numériques, les bases de données scientifiques, les systèmes de commerce en ligne et les systèmes de réservation de voyages sont aujourd'hui disponibles grâce à des interfaces d'interrogation en ligne sur internet. La recherche d'information dans les bases de données (BD) est de plus en plus sollicitée par de simples utilisateurs. Cependant, compte tenu du modèle de sélection des réponses, souvent basé sur un appariement exact, du volume d'information de plus en plus important, et de l'imprécision dans l'expression de son besoin, l'utilisateur se trouve souvent confronté aux problèmes de réponses vides (*Empty Answer Problem*) lorsque sa requête est très sélective ou de réponses nombreuses (*Overbundant Answer Problem*) dans le cas de requête peu sélective (Agrawal *et al.,* 2003) (Samyr *et al.,* 2014) (Premkumar *et al.,* 2014).

Dans cet article, on s'intéresse plus particulièrement à la seconde problématique, qui apparait lorsque l'utilisateur n'a pas une idée précise de son besoin d'information et dont l'expression est généralement vague ou imprécise. Dans une telle situation, il pose une requête large et peu sélective pour éviter l'exclusion des résultats potentiellement intéressants. En réponse à une telle requête, le système peut retourner un nombre important de tuples dont l'exploitation peut être fastidieuse. Tous les tuples retournés satisfont toutes les conditions spécifiées dans la requête. Ceci est dû au modèle classique d'interrogation utilisé dans les Systèmes de Gestion de Bases de Données (SGBD). Ces derniers dans leur majorité supportent uniquement un modèle d'interrogation booléen de type tout ou rien (Agrawal *et al.,* 2003) et (Meng *et al.,* 2009) où la notion de pertinence n'existe pas. La correspondance est exacte, les données répondent complètement à une requête ou ne répondent pas.

Pour départager les tuples résultats, une solution est d'aller regarder les attributs non spécifiés. Cependant le traitement de ce type d'attributs est particulièrement difficile puisqu'on ne connait pas « a priori » quelles sont les préférences de l'utilisateur (Chakrabarti *et al.,* 2004).

La mesure de la pertinence d'une ressource (un document par exemple) vis-à-vis d'une requête et le classement des résultats par ordre décroisant de cette pertinence sont au cœur des préoccupations de la RI. Une panoplie de modèles de RI a été proposée, on en trouve une bonne partie d'entre eux dans l'ouvrage de (Baeza-Yates *et al.,* 2011). Certains ont été adaptés pour répondre à cette problématique de tri de réponse en BD, comme le modèle vectoriel (Agrawal *et al.,* 2003)) et le modèle probabiliste de base (Chaudhuri *et al.,* 2004). Ces modèles considèrent un tuple comme un document.

Dans cet article, on considère également qu'un tuple correspond à un document et on propose une approche basée sur l'adaptation du modèle de langue (ML) de la

RI (Ponte *et al.,* 1998) pour trier les résultats retournés selon leur pertinence vis-à-vis de la requête. Pour estimer le score de pertinence nous utilisons un modèle de langue bi-gramme modélisent la distribution des bi-valeurs d'attributs dans un tuple. Plus précisément, ces probabilités sont estimées à travers les fréquences d'cooccurrence des valeurs des attributs spécifiés et non spécifiés dans la requête avec un lissage par interpolation (Hiemstra, 1998) qui combine linéairement le modèle de l'ensemble des résultats et de la base de données.

La fonction de calcul de score de pertinence proposée prend en considération les différences afférentes aux types de conditions utilisées dans la requête qui peuvent être soient atomiques « $A_i = q_i$ », soient de type BETWEEN (pour les attributs numériques) ou IN (pour les attributs catégoriels). En effet, dans le cas de condition de type BETWEEN et IN, les tuples retournés contiennent des valeurs différentes dans les attributs spécifiés en plus des différences dans les valeurs des attributs non spécifiés. A cet effet, nous proposons deux variantes pour calculer le score de pertinence locale des valeurs des attributs selon le type de condition.

Il convient de noter que dans le cadre du tri des résultats des bases de données, la notion de pertinence n'est pas identique à celle connu en RI, ici elle correspond à une préférence utilisateur puisque tous les tuples de l'ensemble des résultats retournés par le SGBD satisfont les conditions de la requête.

Le reste de cet article est organisé comme suit : dans la section 2, un aperçu sur quelques travaux de recherche qui traitent le problème de réponses nombreuses est donné. Nous présentons ensuite notre approche dans la section 3. Une évaluation expérimentale de l'approche proposée avec une discussion sont présentées respectivement dans les sections 4 et 5. On termine ensuite par une conclusion.

2. Etat de l'art

Pour remédier à cette problématique et faciliter l'exploitation voire l'exploration des réponses nombreuses, trois catégories de solution ont été proposées dans la littérature. La première propose une reformulation de la requête initiale où les prédicats des conditions vagues sont réduits automatiquement pour obtenir un nombre restreint de réponses (Bosc *et al.,* 2006) (Bosc *et al.,* 2008) (Samyr *et al.,* 2014). La deuxième catégorie (Bechchi *et al.,* 2010) (Chen *et al.,* 2004) (Chakrabarti *et al.,* 2004) suggère la classification des résultats retournés et leur présentation sous forme d'une arborescence de navigation qui facilite leur exploration. La troisième catégorie consiste à trier les résultats selon leur degré de pertinence vis-à-vis de la requête en adoptant les techniques de la Recherche d'Information classique (RI) (Agrawal *et al.,* 2003) (Chaudhuri *et al,* 2004) (Su *et al,* 2006) et (van Bunningen *et al,* 2007).

Plusieurs modèles de la RI ont été adaptés pour répondre au besoin de tri en base de données, on en trouve par exemple une adaptation du modèle vectoriel (Agrawal

et al., 2003), deux adaptations du modèle probabiliste (Chaudhuri *et al.,* 2004) (Chaudhuri *et al.,* 2006) et (van Bunningen *et al.,* 2007) et l'approche QRRE « Query Result Ranking for E-commerce» (Su et al, 2006). Dans tous ces modèles, un tuple est considéré comme un document représenté par l'ensemble des valeurs de ses attributs. La table entière ou la BD représente la collection de documents.

Dans (Chaudhuri *et al.,* 2004) (Chaudhuri *et al.,* 2006), les auteurs proposent l'adaptation de modèle probabiliste de base de la RI. Le choix de ce modèle a été motivé par sa capacité à traiter les dépendances existantes entre les valeurs des attributs, contrairement à d'autres modèles tels que le modèle vectoriel.

L'approche proposée utilise l'historique des requêtes pour détecter les relations de dépendance entre les valeurs des attributs spécifiés et les attributs non spécifiés dans la requête. La fonction de calcul du score de pertinence proposée s'exprime comme suit :

$$\text{score}(T, Q) \propto \prod_{y \in Y_Q} \frac{P(y|W)}{P(y|D)} \prod_{y \in Y_Q} \prod_{x \in X_Q} \frac{P(x|y, W)}{P(x|y, D)}$$

Où, D est une table de la base de données ayant n tuples $\{t_1,\ldots,t_n\}$ et un ensemble A de m attributs $A= \{ A_1,\ldots,A_m \}$. $X_Q = \{X_1, X_2, \ldots, X_s\} \subseteq A$ et $Y_Q= A\text{-}X_Q$ représentent respectivement l'ensemble des attributs spécifiés et non spécifiés dans la requête Q. W est l'historique des requêtes « workload ».

Dans cette fonction, le $\text{score}(T)$ est composé de deux facteurs. Le premier estime l'importance globale du tuple tandis que le second évalue sa pertinence conditionnelle. Les probabilités $P(y|W)$ et $P(y|D)$ sont évaluées par la fréquence de la valeur y dans l'historique et dans la base de données respectivement. $P(y|D)$ est similaire à IDF (inverse document frequency) utilisé dans la RI. Les probabilités $P(x|y, W)$ et $P(x|y, D)$ sont les probabilités de x sachant y dans l'historique des requête W et dans la table D, elles sont estimées par les fréquences des cooccurrences des valeurs x et y. Ces probabilités sont précalculées et sauvegardées dans une table auxiliaire ce qui représente une des limités de cette approche notamment à cause du problème de leurs mises à jour suivant l'évolution des données de la BD. Une autre limitation de cette approche est l'utilisation de l'historique des requêtes W qui n'est pas toujours disponible.

Dans QRRE (Su *et al.,* 2006) la fonction d'évaluation du score de pertinence combine l'importance des valeurs des attributs avec le degré de préférence des attributs. Ces facteurs sont estimés par des distributions de probabilité des valeurs des attributs dans la base de données et dans l'ensemble des tuples résultat. Cette approche se base sur l'attribut « prix » dans l'évaluation du degré de préférence des valeurs des autres attributs ce qui fait qu'elle est applicable uniquement dans les BD commerciales. Cette dépendance à un attribut constitue la limite majeure de cette approche.

Dans (Agrawal *et al.,* 2003) les auteurs proposent une adaptation du modèle vectoriel. Les requêtes et les tuples sont représentés dans l'espace de tous les attributs. Le score de pertinence entre une requête Q : $(q_1, q_2, \ldots., q_k)$ et un tuple T : $(t_1, t_2, \ldots., t_k)$ est évalué par:

$$\text{Score}(T, Q) = \sum_{k=1}^{m} S_k(t_k, q_k)$$

Où $S_k(t_k, q_k)$ est le coefficient de similarité entre la valeur q_k de l'attribut A_k spécifiée dans la requête et t_k la valeur afférente au même attribut dans le tuple T. Il est estimé par :

$$S_k(t, q) = \begin{cases} QF_k(q) * IDF_k(q) & \text{si } t = q \\ 0 \text{ sinon} \end{cases}$$

$QF_k(q) * IDF_k(q)$ est une adaptation de la pondération TF-IDF utilisée dans la RI. $IDF_k(q)$ représente l'importance inverse de la valeur q de l'attribut A_k dans la base de données. Elle est évaluée selon la nature de l'attribut (catégorique ou numérique). $QF_k(q)$ représente l'importance de la valeur q dans l'historique des requêtes. Plus il y a des requêtes dans l'historique ayant la valeur q dans l'attribut A_k plus cette valeur est importante.

Une seconde catégorie d'approches suggère la classification des résultats. On cite ici les travaux de (Chakrabarti *et al.,* 2004) et (Bechchi, 2010). Dans (Chakrabarti *et al.,* 2004) une méthode de construction automatique d'une arborescence de navigation des réponses basée sur le contenu des tuples dans l'ensemble des résultats a été proposée. A chaque niveau, la répartition se fait sur la base d'un attribut unique, qui reste le même pour tous les nœuds de ce niveau. Un attribut est utilisé une seule fois comme facteur de classement. L'ordre dans lequel les attributs sont présentés dans l'arborescence et les valeurs utilisées pour éclater les domaines des attributs afférents sont inférés à travers l'exploitation de l'historique des requêtes. Les attributs qui apparaissent le plus dans les anciennes requêtes sont présentés dans les premiers niveaux.

Dans (Bechchi, 2010) l'auteur propose l'algorithme ESRA « *Explore Select Rearrange Algorithm* » qui opère sur les résumés des données au lieu des données elles-mêmes. Il utilise les résumés SAINTETIQ (Raschia *et al.,* 2002) (Saint-Paul *et al.,* 2005) pré-calculés sur l'ensemble des données pour regrouper les réponses d'une requête utilisateur en un ensemble de classes organisées hiérarchiquement.

Un autre type d'approche consiste à réduire automatiquement la portée des conditions de la requête de telle sorte qu'elle puisse retourner un nombre réduit de réponses en rapport avec la requête originale (Bosc *et al.,* 2006) (Bosc *et al.,* 2008) (Samyr *et al.,* 2014). Dans (Samyr *et al.,* 2014) les auteurs proposent une transformation basée sur le concept de proximité sémantique des prédicats flou (Hadj Ali *et al.,* 2003) qui permet de traiter les deux problématiques afférentes aux

réponses nombreuses et réponses vides et ce en élargissant ou en réduisant les intervalles des conditions selon le cas.

Il convient de noter que le principe de relaxation des conditions a été proposé initialement pour traiter le problème de réponse vide. Il s'agit d'assouplir les contraintes sur les données recherchées vers une forme moins restrictive, de façon à ce que le nouvel ensemble de réponses soit plus grand que l'ensemble original (Amer-Yahia *et al.,* 2002). Dans le cas de réponses nombreuses, (Samyr *et al.,* 2014) utilise les mêmes principes pour restreindre les conditions et réduire ainsi le nombre des réponses retournées.

3. Un modèle de langue pour le tri des réponses nombreuses

3.1. Principe de l'approche proposée

Les modèles de langue en recherche d'information sont basées sur l'hypothèse suivante: « un utilisateur en interaction avec un système de recherche fournit une requête en pensant à un ou plusieurs documents qu'il souhaite retrouver ». La requête est alors inférée par l'utilisateur à partir de ces documents (Ponte *et al.,* 1998). La pertinence d'un document vis-à-vis d'une requête est évaluée par la probabilité que la requête soit générée par le modèle de langue du document.

Un modèle de langue est défini par « une fonction de probabilité P qui assigne une probabilité P(s) à un mot ou une séquence de mots s dans une langue » (Boughanem *et al.,* 2004). En recherche d'information, les modèles de langue ont été introduits en 1998 par Ponte et Croft (Ponte *et al.,* 1998) qui voient le score d'un document face à une requête comme la probabilité que la requête soit générée par le modèle du document. En pratique et pour simplifier les calculs, il est supposé dans les modèles de langue, qu'un terme t_i ne dépend que de ses i-1 prédécesseurs. Ainsi, les modèles les plus utilisés sont les modèles unigramme, bigramme et trigramme. Dans le modèle bi-gramme, il est supposé qu'un terme t_i ne dépend que de son prédécesseur immédiat t_{i-1}. Ainsi, la probabilité d'une séquence S= $t_1, t_2, \ldots . t_n$ est évaluée par (Song et al., 1999):

$$P(S) = \prod_{i=1}^{n} P(t_i|t_{i-1}) = \prod_{i=1}^{n} \frac{P(t_{i-1}t_i)}{P(t_{i-1})} \quad [1]$$

Par analogie à la RI, on considère que lorsqu'un utilisateur fait une recherche dans une base de données, il a en tête le ou les tuples idéaux à son besoin d'information. Il traduit ces tuples sous forme d'une requête dans laquelle il spécifie les valeurs des attributs qu'il souhaite. En général la requête ne contient pas tous les attributs du tuple. En effet, l'utilisateur n'est pas censé connaître le schéma détaillé

de la base de données mais il peut avoir connaissance uniquement des attributs les plus importants où fournis par l'interface web.

Le principe de notre approche est le suivant : on considère que le score de pertinence d'un tuple T retourné par le SGBD en réponse à une requête Q est défini par la probabilité de son appartenance au modèle de langue des bonnes réponses (M_{brQ}) où $brQ \subset R$: l'ensemble des réponses retournés par le SGBD. Ainsi, le calcul de la pertinence du tuple T vis-à-vis d'une requête Q revient donc à mesurer la vraisemblance, la probabilité que la requête Q soit générée par le modèle langue des bonnes réponses : $\mathrm{Score}(Q, T) = P(T|M_{brQ})$

Pour estimer la probabilité $P(T|M_{brQ})$, il est possible de faire des hypothèses simplificatrices. Par exemple, on peut supposer que les valeurs des attributs sont indépendantes les unes des autres (un sac de valeurs d'attributs) comme c'est le cas dans la recherche d'information textuelle (un sac de mots). Cependant, dans les bases de données, il y a une dépendance fonctionnelle entre les valeurs des attributs (Chaudhuri *et al.,* 2004).

Dans notre approche, on suggère l'utilisation d'un modèle de langue bi-gramme pour estimer la probabilité $P(T|M_{brQ})$. Ce choix est motivé d'une part par sa simplicité et d'autre par sa capacité de capter une partie des dépendances existantes entres les valeurs d'attribut deux à deux. De plus avec les modèle de langue on n'aura pas besoins ni d'une phase de prétraitement pour estimer les probabilités ni de l'historique de requêtes.

3.2. Score de pertinence des tuples vis-à-vis de la requête

Soit D une table d'une BD ayant m attributs, $A = \{A_1, A_2, \dots, A_m\}$.

Soit R l'ensemble des tuples résultats retournés en réponse à une requête conjonctive $Q = \{q_1, q_2, \dots, q_n\}$ où n est le nombre d'attribut spécifiés dans la requête avec $n \leq m$. Chaque $T \in R$ est représenté par les valeurs des attributs $T = \{a_1, a_2, \dots, a_m\}$.

Pour différencier entre les attributs spécifiés et non spécifiés dans la requête, on définit $S = \{A_1, A_2, \dots, A_n\} \subseteq A$ et $NS = A - S$ comme étant respectivement l'ensemble des attributs spécifiés et l'ensemble des attributs non spécifiés dans la requête.

Pour trier les tuples qui satisfont les conditions de la requête, il faut tout d'abord déterminer le(s) critère(s) qui permettent de les départager (différencier). Pour ce faire, nous proposons d'explorer les différences qui existent dans les valeurs des attributs, non spécifiés dans la requête. En particulier, notre intuition porte sur l'exploitation des dépendances existantes entres les valeurs des attributs spécifiés dans la requête et les valeurs des attributs non spécifiés. Les tuples dont les valeurs des attributs non spécifiés ont une forte dépendance avec les valeurs spécifiées dans

la requête seront privilégiés. On suppose que l'utilisateur préfèrerait les tuples ayant des valeurs des attributs qui dépendent de ceux qu'il a spécifiés dans sa requête.

Dans les bases de données, l'ordre entre les valeurs des attributs a_i et a_j n'a pas d'importance. On considère que les valeurs des attributs sont dépendantes entre elles, car elles décrivent le même tuple, mais leur ordre d'apparition n'a aucune influence. On suppose que l'on peut capturer une partie de ces dépendances en prenant les attributs deux à deux. Ainsi, pour adapter l'équation [1] à notre contexte, on considère que le tuple T est une suite des valeurs d'attributs dépendantes les unes aux autres deux à deux où l'ordre n'a aucune importance.. On obtient alors :

$$score(Q,T) = P(T|M_{br}) = \prod_{i=1}^{m} \prod_{j=i+1}^{m} P(a_j|a_i) = \prod_{i=1}^{m} \prod_{j=i+1}^{m} \frac{P(a_i, a_j)}{P(a_i)} \qquad [2]$$

Où $P(a_j|a_i)$ représente la probabilité que l'attribut A_j ait la valeur a_j sachant que l'attribut A_i a la valeur a_i.

Dans l'équation [2] on traite le cas général dans lequel on ne fait pas de distinction entre les attributs spécifiés et les attributs non spécifiés dans la requête. Cette distinction sera discutée dans les sections 3.2.1 et 3.2.2.

La notion de dépendance entre deux valeurs des différents attributs est évaluée selon la distribution des données dans l'ensemble R et la table D. Elle se traduit par la fréquence de leur apparition dans les mêmes tuples dans l'ensemble R et dans la table. Plus la fréquence d'occurrence dans les mêmes tuples des deux valeurs est grande plus leur dépendance est forte.

Pour évaluer $P(a_i, a_j)$ on utilise un lissage avec interpolation qui permet d'assigner des probabilités non nulles. On a opté pour un lissage par modèle de la collection (Hiemstra, 1998). La $P(a_i, a_j)$ est mesurée par une combinaison linéaire du modèle de l'ensemble des résultats R et du modèle la table D. On obtient alors :

$$P(a_i, a_j) = \beta * P_{ML}(a_i, a_j|R) + (1 - \beta)\, P_{ML}(a_i, a_j|D)$$

$P_{ML}(a_i, a_j|R)$ (respectivement $P_{ML}(a_i, a_j|D)$) estime la dépendance entre les deux valeurs a_i et a_j dans l'ensemble R (respectivement dans la table D). Le paramètre β est une constante fixée de manière expérimentale.

Les probabilités $P_{ML}(a_i, a_j|R)$ et $P_{ML}(a_i, a_j|D)$ sont estimées en utilisant l'estimation de vraisemblance maximale (*Maximum Likelihood, ou ML*) et ce comme suit :

$$P_{ML}(a_i, a_j|R) = \frac{f_r(a_i,a_j)}{|R|} \quad \text{et} \quad P_{ML}(a_i, a_j|D) = \frac{f_d(a_i,a_j)}{|D|}$$

Où :

- $f_r(a_i, a_j)$ (resp $f_d(a_i, a_j)$) représente le nombre de tuples dans l'ensemble R (resp dans la table D) ayant la valeur a_i pour l'attribut A_i et la valeur a_j pour l'attribut A_j ;

- |R| (resp|D|) est le nombre de tuples dans l'ensemble R (respectivement dans la table D).

La probabilité $P(a_i)$ est estimée de la même manière que $P(a_i, a_j)$:

$$P(a_i) = \beta \frac{f_r(a_i)}{|R|} + (1 - \beta) \frac{f_d(a_i)}{|D|}$$

La question qui se pose à cette étape de la définition du modèle porte sur les attributs à prendre en compte dans l'estimation des probabilités $P(a_i, a_j)$. Ceci dépend du type des conditions spécifiés dans la requête ; conditions atomiques de type ($A_i = q_i$) et conditions de type BETWEEN (pour les attributs numériques) ou IN (pour les attributs catégoriels). Nous discutons ces deux cas dans ce qui suit.

3.2.1. Cas de requête avec des conditions atomiques

Dans le cas des requêtes avec des conditions atomiques de type ($A_i = q_i$), les valeurs des attributs spécifiés sont les mêmes pour tous les tuples retournés. Par conséquent, les probabilités $P(a_i, a_j)$ sont les mêmes pour tous les tuples de l'ensemble R. De ce fait, au lieu de considérer que les valeurs des attributs sont toutes dépendantes les unes aux autres deux à deux, on opte pour une solution intermédiaire, déjà utilisée par (Chaudhuri *et al.,* 2004). On s'intéresse à la dépendance entre les attributs spécifiés et non spécifies. On exploite uniquement les dépendances partielles entres attributs de l'ensemble S et ceux de l'ensemble NS.

Par ailleurs, on note aussi que tous les tuples retournés ont les mêmes valeurs $a_i = q_i$ dans les attributs spécifiés ($A_i \in S$). Ainsi $P(a_i)$ est la même pour tous les tuples résultats. Par conséquent, $P(a_i)$ n'a aucun impact sur le résultat du tri, elle sera donc ignorée.

L'équation [2] peut s'écrie alors comme suit :

$$score(Q, T) = \prod_{a_i \in S} \prod_{a_j \in NS} P(a_j | a_i) \propto \prod_{a_i \in S} \prod_{a_j \in NS} P(a_i, a_j) \quad [3]$$

La probabilité $P(a_j | a_i)$ représente le degré de dépendance de la valeur a_j de l'attribut A_j non spécifié sachant que la valeur $a_i = q_i$ est spécifiée dans la requête Q. Si on remplace dans l'équation [3] a_i par q_i « puisque $a_i = q_i$ » et $P(a_i, a_j)$ par son estimation, on obtient :

$$\text{score}(Q,T) \propto \prod_{i=1}^{n} \prod_{a_j \in NS} \left(\beta \frac{f_r(q_i, a_j)}{|R|} + (1-\beta) \frac{f_d(q_i, a_j)}{|D|} \right)$$

Pour illustrer notre approche, on va suivre l'exemple suivant : supposant la requête avec des conditions atomiques suivante : $Q = \{q_1, q_3\}$ sur une table ayant cinq attributs $A = \{A_1, A_2, A_3, A_4, A_5\}$. Alors, le score de pertinence d'un tuple $T = \{a_1 = q_1, a_2, a_3 = q_3, a_4, a_5\}$ résultat vis-à-vis de la requête est calculé comme suit:

$$\text{score}(Q,T) = P(a_1, a_2)P(a_1, a_4)P(a_1, a_5)P(a_3, a_2)P(a_3, a_4)P(a_3, a_5)$$

Les probabilités $P(a_1, a_3)$, $P(a_1)$ et $P(a_3)$ ne sont pas prises en compte parce qu'elles ont les mêmes valeurs pour tous les tuples résultats, elles n'influent donc pas sur le tri.

3.2.2 Cas de requête avec des conditions BETWEEN et IN

Une forme plus générale des requêtes est celle ayant des conditions de type A_i BETWEEN $(q_{i\,min}, q_{i\,max})$ pour les attributs numériques et A_i IN $\{q_{i1}, \dots, q_{i\,l}\}$ pour les attributs catégoriels. Dans ce contexte, les tuples résultats contiennent des valeurs différentes dans les attributs spécifiés.

Pour illustrer l'importance de ces différences, considérant à titre d'exemple une requête Q : kilométrage BETWEEN (50000,150000) AND (année = 2009) sur une table de véhicule d'occasion. Dans cet exemple, il est plus probable que les voitures ayant un kilométrage proche de 50000 Km soient plus intéressantes pour l'utilisateur par rapport à celles ayant un kilométrage proche de 150000 Km. Cependant, cette hypothèse peut être mise en cause vu que les voitures anciennes ayant un kilométrage faible ont un prix élevé, donc moins intéressantes par rapport aux autres voitures datant de 2009.

A travers l'exemple susmentionné, on remarque que pour trier de manière efficace les tuples ; contrairement aux requêtes avec des conditions atomiques, on ne peut pas ignorer les dépendances qui existent entre les attributs spécifiés. Donc, tous les attributs de la table seront considérés.

Pour adapter l'équation [2] à ce type de condition BEWTEEN et IN on note que:

– Les probabilités $P(a_i)$ ne sont pas identiques pour tous les tuples lorsque l'attribut $A_i \in S$ est spécifié par une condition de type BETWEEN ou IN. Par conséquent, elles doivent être prises en considération ;

– La préférence d'un tuple pour un utilisateur change en fonction des valeurs des attributs spécifiés. Pour cela, on doit prendre en considération les dépendances entre les valeurs des attributs spécifiés en plus des dépendances entre les valeurs des attributs spécifiés et non spécifiés.

Ainsi, dans le cas où la requête est composée uniquement des conditions de type BETWEEN et IN, la fonction de calcul du score de pertinence peut s'exprimer par :

$$\text{score}(Q,T) = \prod_{a_i \in S} \prod_{\substack{a_j \in A \\ a_j \neq a_i}} \frac{P(a_i, a_j)}{P(a_i)} \qquad [4]$$

3.2.3 Forme générale de la fonction du calcul du score de pertinence

De ce qui précède, il ressort que le score de pertinence global dépend étroitement des types des conditions utilisés dans la requête. Dans le cas d'une requête contenant les deux types de conditions (atomique, BETWEEN et IN). Le score de pertinence est calculé en combinant les fonctions [3] et [4] et ce, selon le type de la condition afférente à chaque attribut spécifié dans la requête.

Si on considère que $C = \{C_1, C_2, \dots, C_n\}$ est l'ensemble des conditions spécifiées dans la requête, alors la forme générale de la fonction de calcul du score de pertinence peut s'exprimer de la manière suivante :

$$\text{score}(Q,T) = \prod_{a_i \in S} \begin{cases} \prod_{a_j \in NS} P(a_i, a_j) & \text{si } C_i : (a_i = q_i) \\ \frac{1}{P(a_i)} \prod_{\substack{a_j \in A \\ a_j \neq a_i}} P(a_i, a_j) & \text{si } C_i : a_i \in [q_{i1}, .., q_{im}] \text{ où } a_i \in \{q_{i1}, .., q_{im}\} \end{cases}$$

4. Evaluation expérimentale

L'évaluation d'un système de recherche d'information s'appuie essentiellement sur l'estimation de la qualité des informations retrouvées par ce dernier. Dans le cas du traitement du problème des réponses nombreuses dans les bases de données, on s'intéresse plus particulièrement à la capacité du système proposé à mettre en début de liste les tuples les plus pertinents qui satisfait l'utilisateur vis-à-vis de sa requête.

Pour évaluer l'efficacité de notre approche, nous l'avons implémentée et expérimentée sur une table d'une base de données de voitures d'occasion extraite à partir du site web (*www.ouedkniss.com*). L'objectif escompté à travers cette évaluation expérimentale est de démontrer l'impact du modèle de langue bi-gramme sur le tri des résultats nombreuses en base de données et par la même évaluer la qualité du tri de notre approche.

Afin d'effectuer cette évaluation, nous avons suivi le même protocole d'évaluation et nous avons adopté la même métrique « la précision moyenne» que celle utilisée par (Chaudhuri et al, 2004) (Chaudhuri *et al.,* 2006) pour évaluer le modèle probabiliste.

4.1. Protocole d'évaluation

Nous avons demandé à cinq (5) personnes de soumettre trois (3) requêtes chacun sur la table « véhicule » selon leur préférence. Cette table contient 16842 tuples, elle est composée des attributs suivants : «*id, marque, modèle, version, année, transmission, énergie, kilométrage, puissance, options, couleur* » dont deux sont numériques « *année et kilométrage* » et les autres attributs sont catégoriels.

Chaque requête avait en moyenne 2,4 attributs spécifiés. Le nombre moyen des tuples résultats retournés en réponse aux 15 requêtes est de 217 tuples, avec un maximum de 494 tuples et un minimum de 92 pour une table ayant 16842 tuple. Ces chiffres illustrent bien que les réponses nombreuses est un problème récurent lors d'une recherche dans une base de données. Le tableau N°1 présente cinq requêtes parmi les 15 (une requête pour chaque utilisateur).

N°	Requête	Tuples retournés
Q3	Modèle IN{ 207, 308}, kilométrage <=50000 année BETWEEN (2009 , 2011)	324
Q4	Marque IN {Peugeot, Renault}, Modèle IN{ 308, Megan 3}, année BETWEEN (2008 , 2011)	215
Q7	Modèle IN{ Symbol, Clio classic},année BETWEEN (2007 , 2010), energie = Essence	202
Q14	Modèle IN{Clio, Clio3}, année BETWEEN(2004 , 2009)	494
Q15	Modèle = 207, année >= 2007, kilométrage <=100000	488

Tableau 1. *Exemple de requêtes du protocole d'évaluation.*

Il est très difficile de demander aux utilisateurs de trier ; pour chaque requête, tous les tuples retournés selon leur pertinence. Pour pallier cette difficulté, nous avons adopté la stratégie suivie par (Chaudhuri *et al.,* 2004) et (Su *et al.,* 2006) qui consiste en: pour chaque requête Q_i, nous avons généré une liste de 30 tuples susceptibles de contenir un bon mélange de tuples «pertinent» et «non pertinents» vis-à-vis de cette requête . On rappelle que la notion de pertinence ici correspond à une préférence utilisateur). La liste des 30 tuples est composée des 10 premiers tuples retournés par notre approche et les 10 premiers tuples retourné par le SGBD (sans aucun tri) ; avec suppression des tuples en double, auxquels il a été ajouté dix tuples choisis aléatoirement. Avant d'être présentés à l'utilisateur, les 30 tuples sélectionnés ont été triés aléatoirement.

Ensuite, nous avons présenté les requêtes avec les listes correspondantes à chaque utilisateur. Leur rôle était de choisir les dix (10) meilleurs tuples qu'ils préfèrent le plus; qualifiés de pertinents, parmi les 30 tuples des listes afférentes à chaque requête. Les autres tuples sont considérés non pertinents. Enfin, nous avons appliqué notre fonction de calcul de score de pertinence et comparé les deux résultats pour chaque requête de test.

Pour évaluer l'impact de notre approche, nous avons adopté comme ligne de base la liste par défaut retournée par le SGBD sans aucune fonction de tri. De plus, nous avons implémenté une fonction de tri aléatoire « RANDOM » des résultats retournés.

4.2. Résultats expérimentaux

Les résultats présentés ci-dessous ont été obtenus avec la valeur du paramètre $\beta = 0{,}8$ pour laquelle nous avons eu les meilleurs résultats sur l'ensemble des expérimentations effectuées.

4.2.1. La qualité du tri des tuples retournés

Pour évaluer la qualité du tri des résultats retournés, nous avons utilisé les mesures de précision / rappel. Nous utilisons la précision à 10 (P@10) et la précision moyenne MAP (Median Average Precision), la moyenne des précisions obtenues à chaque occurrence d'un tuple pertinent dans la liste triée. La précision est le ratio entre le nombre de tuples pertinents retournés et le nombre total de tuples retournés. La précision à 10 tuples (P@10) est celle calculée pour les dix premiers tuples. On rappelle ici que les utilisateurs ont été invités à choisir les dix (10) meilleurs tuples qu'ils préfèrent le plus. Ainsi, la P@10 nous permet d'évaluer la façon dont la préférence de l'utilisateur est capturée par notre approche. La figure n°1 illustre les P@10 obtenues par les trois approches évaluées.

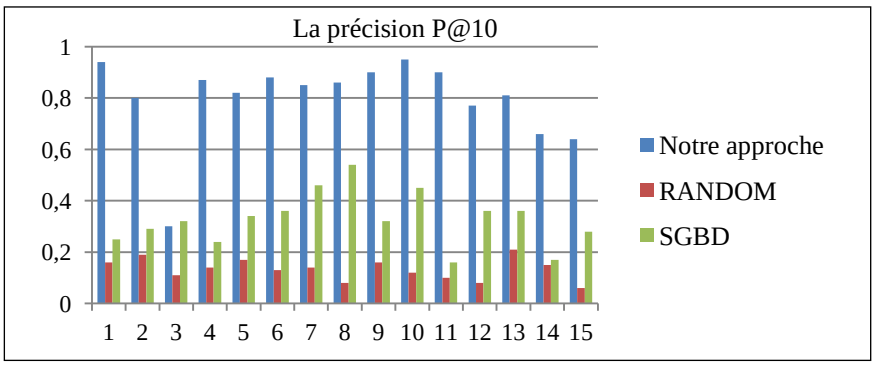

Figure 1. *La précision P@10 obtenue pour chaque requête.*

Ces résultats préliminaires montrent que l'approche proposée permet en effet une amélioration des résultats de 14 requêtes sur 15. Pour la requête n°3 qui présente un résultat différent, la P@10 de notre approche est légèrement inferieure à celui du SGBD. Cela est dû au fait que dans les 324 tuples résultats il n'y a pas beaucoup de différence dans les valeurs des attributs non spécifiés ce qui ne permet pas de bien les départagés automatiquement.

4.2.2. Le moyenne des précisions moyennes

La moyenne des précisions moyennes MAP obtenue par notre approche est de 0,79 suivie de 0,32 et de 0,13 respectivement pour la liste par défaut du SGBD et le tri aléatoire.

	Notre approche	**SGBD**	**RANDOM**
MAP	**0,79**	0,32	0,13

Tableau 2. *La moyenne des précisions moyennes.*

5. Discussion

Les travaux les plus proches au notre sont l'adaptation du modèle de probabiliste de base de la RI (Chaudhuri *et al.,* 2004) (Chaudhuri *et al.,* 2006) et la proposition du (Su *et al.,* 2006). Les avantages principaux de notre approche par rapport à ces deux travaux portent principalement sur :

- Phase de prétraitement : dans notre approche le score de pertinence est calculé en une seule phase. Il n'y a pas de phase de pré traitement. Par contre, l'approche QRRE (Su *et al.,* 2006) requière une phase de pré traitement afin d'estimer les histogrammes (mono-attribut et bi-attribut) de fréquence d'occurrence des différentes valeurs des attributs. Presque la même démarche est adoptée dans la phase de pré traitement de l'approche de (Chaudhuri *et al.,* 2004). Les probabilités atomiques qui composent la fonction d'évaluation du score de pertinence sont calculées et sauvegardées dans des tables auxiliaires ;

Par ailleurs, l'inconvénient majeur des phases de prétraitement est la mise à jour des données estimées avec l'évolution des données de la base données.

– L'historique des requêtes : notre approche ne se base pas sur l'historique des requêtes, qui n'est pas toujours disponible, comme c'est le cas dans l'adaptation du modèle probabiliste proposée dans (Chaudhuri *et al.,* 2004) ;

– Indépendance du domaine : notre approche est indépendante du domaine, le calcul du score de pertinence ne dépend pas d'un attribut spécifique parmi les attributs de la table comme c'est le cas dans l'approche QRRE (Su *et al.,* 2006). Dans cette dernière, l'attribut « Prix » joue un rôle principal dans l'estimation du degré d'importance des valeurs des attributs. Cette dépendance vis-à-vis de l'attribut « Prix » fait que l'approche QRRE est applicable uniquement dans les bases de données commerciales.

6. Conclusion

Cet article présente une adaptation du modèle de langue de la recherche d'information pour le tri des tuples dans les bases de données. Le but est de remédier à un problème récurrent en base de données en l'occurrence, les réponses nombreuses. Le principe de notre approche est basé sur le calcul du score de pertinence des tuples résultats vis-à-vis de la requête en s'appuyant sur les modèles statistiques de langue.

Le modèle de langue choisi est le modèle bi-gramme. Ce choix est motivé par sa capacité à capter les dépendances entre les valeurs des attributs spécifiés dans la requête et ceux qui ne le sont pas. Le modèle proposé se base sur l'hypothèse que l'utilisateur préfèrerait les tuples ayant des valeurs des attributs non spécifiés fortement dépendantes des valeurs spécifiées dans sa requête.

Nous avons évalué notre approche en adoptant le même protocole d'évaluation que celui suivi par les deux travaux connexes. Les résultats obtenus par notre approche en termes de qualité de tri des résultats retournés et de capture des préférences montrent que trier les tuples en exploitant les dépendances des valeurs des attributs spécifiés dans la requête et ceux qui ne le sont pas, est bénéfique pour l'utilisateur.

Bibliographie

Agrawal, S., Chaudhuri, S., Das, G., Gionis, A., «Automated Ranking of Database Query Results», *CIDR Conference*, 2003.

Amer-Yahia, S., C., Srivastava, D., « Tree Pattern Relaxation », *EDBT'02*, Prague, Czech Republic, p. 496-513, 2002.

Baeza-Yates Ricardo A., Ribeiro-Neto Berthier A., *Modern Information Retrieval: the concepts and Technology behind search (2nd edition)* ACM Press, 2011.

Bechchi Mounir., Clustering-based Approximate Answering of Query Result in Large and Distributed Databases. Thèse doctorat en informatique, Université de Nantes, 2010.

Boughanem, M. Kraaij W., Nie J.Y., « Modèles de langue pour la recherche d'informations », *Les systèmes de recherche d'informations - Modèles conceptuels* , ed.Hermes, pp. 163-184, 2004.

Bosc, P., Hadjali, A., Pivert, O., «About Overabundant Answers to Flexible Queries » *acte des 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems IPMU'06,* Paris, 2-7 July 2006, p. 2221-2228.

Bosc, P., Hadjali, A., Pivert, O., « Empty versus overabundant answers to flexible queries» *Fuzzy Sets and Systems Journal*, vol. 159, n°12, 2008, p.1450-1467.

Chakrabarti, K., Chaudhuri, S., and Hwang, S. «Automatic categorization of query results». *In Proceedings of the SIGMOD Conference*, p. 755–766, 2004.

Chaudhuri S, Das G, Hristidis V, Weikum G, « Probabilistic Ranking of Database Query Results »,*Proceedings of the 30th VLDB Conference*, Toronto, Canada, 2004.

Chaudhuri, S., Das, G., Hristidis, V. « Probabilistic information retrieval approach for ranking of database query results. ». *ACM Trans. Database Syst.*, vol. 31, n°3, 2006, p.1134–1168.

Chen Z. Y, Li. T, Addressing diverse user preferences in SQL-Query-Result navigation. In Proceedings of the SIGMOD Conference,2007, p. 641-652.

Hadj Ali A., Dubois D, Prade H., «Qualitative Reasoning Based on Fuzzy Relative Orders of Magnitude », *IEEE Transactions on Fuzzy Systems*, vol. 11, n°1, 2003, p. 9-23. http://dx.doi.org/10.1109/TFUZZ.2002.806313

Hiemstra, D., « A linguistically motivated probabilistic model of information retrieva l », *European Conference of Digital Library (ECDL98)*, Sept. 1998, Springer Verlag.

Meng, X., Ma, Z., M., Yan L., « Answering Approximate Queries over Autonomous Web Databases », *Track: XML and Web Data / Session: XML Querying*, 2009 MADRID.

Ponte, J., M., Bruce, W., Croft, A., « Language Modeling Approach to Information Retrieval », *Research and Development in Information Retrieval, ACM-SIGIR*, 1998, p. 275-281.

Premkumar, S., Gayathri, K., Raghuram, I.S., « Estimation of Ranking Score for Database Query Results », *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, n° 6, June 2014, p. 1108-1115.

Raschia G. , Mouaddib N., « SAINTETIQ: a fuzzy set-based approach to database summarization», *Fuzzy Sets Syst.*, vol. 129, n°2, 2002, p.137–162.

Saint-Paul R, Raschia G, Mouaddib N., «General purpose database summarization », *In VLDB '05*, 2005, p 733–744.

Samyr A.M., Silvio do Lago P., « Dealing with Empty and Overabundant Answers to Flexible Queries », *Journal of Data Analysis and Information Processing*, 2014, vol. 2, p.12-18

Song Fei., Croft W. Bruce., «A General Language Model for Information Retrieval», *Proceedings of the eighth international conference on Information and knowledge management (CIKM)*, 1999, p. 316-321.

Su Weifeng, Jiying Wang, Qiong Huang, Fred Lochovsky. « Query Result Ranking over E-commerce Web Databases», *CIKM'06,* November 6–*11*, 2006, Arlington, VA Virginia, USA.

VanBunningen A. H., Fokkinga M. M., Peter M.G. Apers « Ranking Query Results using Context-Aware Preferences», IEEE 23rd International Conference on Data Engineering Workshop, 2007.