Recherche d'informations sur le Web basée sur des données d'opinion

Mehdi Adda*, Rokia Missaoui**, Petko Valtchev***

*Département de mathématiques, d'informatique et de génie, Université du Québec À Rimouski
mehdi_adda@uqar.qc.ca,
**Département d'informatique et d'ingénierie, Université du Québec en Outaouais
rokia.missaoui@uqo.ca
***Département d'informatique, Université du Québec À Montréal
valtchev.petko@uqam.ca

Résumé. Avec la diversification du contenu disponible sur le Web, aussi bien en quantité qu'en qualité, retrouver l'information pertinente est problématique. Les moteurs de recherche d'informations sur le Web ont comme objectif de permettre de retrouver efficacement l'information pertinente. Ces moteurs implémentent différentes stratégies pour déterminer la pertinence des pages Web. Ces dernières sont classées en se basant principalement sur des critères d'évaluation automatisés, ce qui crée un vide entre la pertinence effective d'un contenu et la pertinence calculée. Afin de réduire cet écart, nous présentons un cadre d'application de moteur de recherche où la pertinence d'une page Web est calculée à partir des évaluations des internautes du contenu de la page (texte, image, audio, vidéo,...). Pour illustrer l'exploitation du cadre proposé, nous avons développé le moteur de recherche expérimental *SocialSeeker*(Adda, 2008). Finalement, une étude empirique préliminaire du potentiel du prototype est présentée et discutée

1 Introduction

La tâche principale d'un moteur de recherche est d'offrir à ses utilisateurs la possibilité de formuler des requêtes et de leur retourner le contenu qui correspond le mieux à ces requêtes (Adah et al., 1997). Les moteurs de recherche scrutent périodiquement, voire de manière continue, l'Internet pour (ré)indexer des pages Web (Sun et al., 2005). Les stratégies utilisées pour classer ces pages diffèrent d'un moteur à l'autre.

1.1 Stratégies de classement des pages Web

Les grands acteurs sur le marché des moteurs de recherche, tels que *Google* (Google, 2008), *Microsoft* (Microsoft, 2008), et *Yahoo* (Yahoo, 2008), gardent secrets les détails des techniques de classement qu'ils utilisent. Cependant, quelques éléments qui jouent un rôle important dans ce classement sont connus (Ding et Chi, 2003).
En premier lieu, les moteurs de recherche prennent en compte aussi bien le titre d'une page

Web que les mots contenus dans son adresse URL (*Universal Resource Locator*). Deuxièmement, les moteurs de recherche utilisent des programmes, appelés "*bots*" ou "*spiders*", pour scruter la structure des pages Web. Parmi les informations recherchées figurent les balises "*meta-tag*" qui sont invisibles aux utilisateurs, et qui renseignent sur le contenu de la page en question. Troisièmement, la fréquence et le contexte d'apparition de mots dans le contenu d'une page sont pris en compte.
Cependant, ces éléments peuvent être facilement manipulés par les *Webmestres*, ce qui a une incidence sur le classement des pages. Pour réduire cette influence, des facteurs moins susceptibles d'être manipulés sont intégrés. Il s'agit principalement de prendre en considération le trafic généré par une page et le nombre de sites qui pointent vers cette page.

Même avec ces extensions, les moteurs de recherche classiques restent entièrement dépendants d'un classement automatique d'une page Web qui peut ne pas correspondre à la pertinence effective du contenu de la page en question. Une solution à ce problème serait de classer les pages Web suivant les évaluations apportées au contenu. Concrètement, cette pertinence peut être mesurée à partir des évaluations que les utilisateurs apportent au contenu sur des sites spécialisés. Dans cet article, nous présentons l'extension et l'évaluation du travail préliminaire présenté dans (Adda et al., 2008) où nous avons présenté l'idée de la prise en compte des données d'opinion pour la recherche d'information sur le Web.

1.2 Données d'opinion et recherche sur le Web

Avec l'émergence du Web2.0 (Lewis, 2006), de plus en plus d'applications, telles que Dzone (Dzone, 2008), Reddit (Reddit, 2008), Digg (Digg, 2008), Slashdot (Slashdot, 2008), sont développées dans le but de partager du contenu et récolter les évaluations des internautes (Borgatti et Cross, 2003; Hooff et al., 2003). Ces applications peuvent être perçues comme étant : (i) des entrepôts de liens vers des ressources Web, et (ii) une interface d'évaluation et de filtrage communautaire des ressources. Les évaluations des utilisateurs pour un contenu peuvent alors être exploitées pour déterminer la pertinence "sociale/communautaire" d'une ressource (Bojrs et al., 2008).

Actuellement, il existe quelques moteurs de recherche qui tentent d'intégrer les données d'opinion pour déterminer la pertinence d'une page Web (*ex. Mahalo* (Mahalo, 2008), *Dipiti* (Dipiti, 2008) et Chacha (ChaCha, 2008)). Ces moteurs permettent aux utilisateurs d'évaluer les résultats proposés, et le système exploite ces informations pour (ré)évaluer le classement des sites. Une des limites de ces systèmes est la difficulté d'intégrer les données d'opinion déjà disponibles sur le Web. Par conséquent, ils nécessitent la création d'une base de données d'opinion propre au système.

Au lieu de créer une nouvelle base de données d'opinion, nous proposons d'exploiter directement les évaluations déjà disponibles sur le Web pour calculer la pertinence des pages. Aussi, le cadre d'application que nous proposons se compose principalement d'un langage de description des données et des classes d'opinions ainsi que d'un ensemble d'opérations qui s'appliquent à ces éléments (*cf.* section 2).
L'une des difficultés rencontrées est liée à la diversité des formes d'expression et de représentation des opinions. À titre d'exemple, il existe des sites qui acceptent des votes binaires (apprécie, n'apprécie pas), d'autres implémentent des systèmes de votes multi-échelle (*ex.* niveau d'appréciation sur une échelle de 3, 5 ou 10 niveaux).
Afin de résoudre ce problème et pouvoir comparer les valeurs de pertinence obtenues de diffé-

rents sites, nous utilisons une formule de calcul de pertinence qui normalise les valeurs collectées. (*cf.* section 2.4).

Pour illustrer la faisabilité du cadre proposé et évaluer son apport, nous avons implémenté un prototype appelé *SocialSeeker* (*cf.* section 4) qui recherche les pages Web ayant les évaluations les plus favorables par rapport à un sujet donné. Dans la section 4, nous faisons une évaluation de la pertinence des résultats de recherche retournés par *SocialSeeker* en comparaison avec *Google*. Cette évaluation est relativement positive et nous encourage à développer davantage cette idée d'intégration des entrepôts d'opinions dans les moteurs de recherche d'information sur le Web.
En section 5, nous présentons notre conclusion ainsi que les améliorations à apporter au cadre proposé et au prototype *SocialSeeker*.

2 Modèle d'intégration des données d'opinion dans la recherche sur le Web

La figure 1 représente une vue générale de la logique d'intégration des données de rétroaction (*feedback*) dans un moteur de recherche. Dans cette figure, nous présentons les réseaux communautaires (ou réseaux sociaux) comme un exemple d'application de partage de références et de ressources. Ces répertoires, que l'on désignera dans le reste de cet article par *entrepôts d'opinions*, contiennent explicitement et/ou implicitement la rétroaction des utilisateurs (Joachims et al., 2007; Shahabi et Chen, 2003). Or, les critères et fonctionnalités offertes par chaque application peuvent différer. Par conséquent, il est nécessaire d'analyser les formes de rétroaction fournies afin de construire un modèle adéquat de calcul de la pertinence.

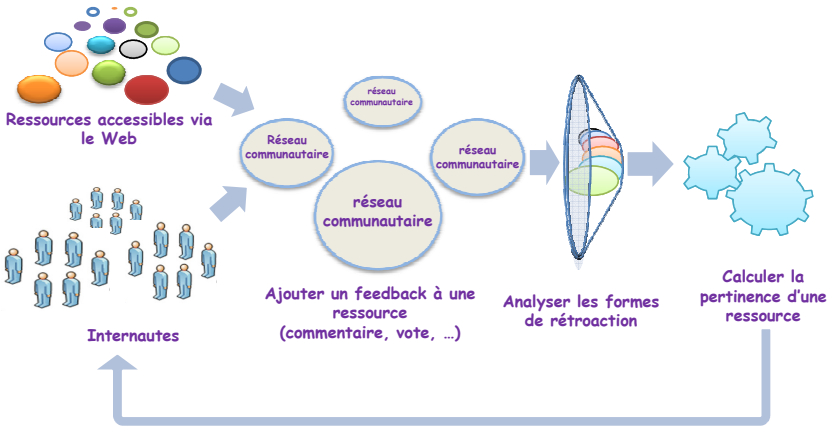

FIG. 1 – *Présentation du processus d'intégration des données d'opinion dans la recherche sur le Web.*

2.1 Scénario type

Afin d'illustrer le fonctionnement d'un moteur de recherche qui se base sur les données de rétroaction, nous présentons dans la figure 2 un scénario type. En premier lieu, l'utilisateur formule une requête. Le système se base sur les mots clés saisis par l'utilisateur pour interroger les différents entrepôts d'opinions disponibles. Par la suite, chaque entrepôt retourne une collection de références à des ressources Web (*ex.* liens, *tags*, résumés, ...) ainsi que les données d'opinion associées à chaque ressource. Le système récupère les résultats et calcule une valeur de pertinence normalisée pour chaque élément suivant les données d'opinion qui lui sont associées. Après classement, les références ayant les plus grandes valeurs de pertinence sont retournées. L'utilisateur peut apporter une (ré)évaluation qui sera directement acheminée à l'entrepôt (*resp.* aux entrepôts) concerné(s).

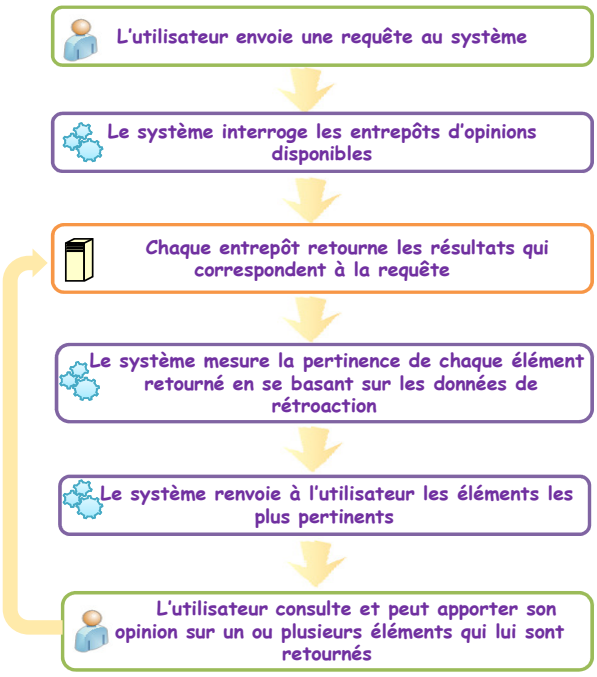

FIG. 2 – *Scénario type d'une recherche basée sur les données d'opinion.*

2.2 Langage de description des données d'opinion

Dans cette section, nous formalisons la description des données d'opinion. Ensuite, nous proposons un langage qui s'inspire de la logique de descriptions (Baader et al., 2003) pour modéliser les classes de ces données. Afin de pouvoir distinguer les aspects purement structuraux des données et la signification de ces structures, le langage est muni d'un ensemble de constructeurs syntaxiques auxquels est associée une sémantique précise. Alors que la partie

Symbole/Relation	Notion représentée
$\mathcal{R}$	Ensemble de ressources accessibles via le Web.
$URLS_i$	Ensemble d'éléments indiquant la (les) localisation(s) de la ressource i dans le Web. $\bigcup_{i \in \mathcal{R}} URLS_i = URLS$.
ϖ_i	Contenu de la ressource i. $\bigcup_{i \in \mathcal{R}} \varpi_i = \varpi$.
χ_i	Ensemble d'informations qui décrivent/complètent la ressource i. $\bigcup_{i \in \mathcal{R}} \chi_i = \chi$.
Ψ_i	Ensemble des données de rétroaction associées à la ressource i. $\bigcup_{i \in \mathcal{R}} \Psi_i = \Psi$.

TAB. 1 – *Notions de base.*

syntaxique définit la structure des classes des données, la partie sémantique leur associe un signification dans le domaine considéré (les évaluations des utilisateurs).

2.2.1 Description des données d'opinion

Les données que nous manipulons sont des enregistrements, appelés *instances d'opinion* (*cf.* définition 2.3), qui sont composés de *références à des ressources Web* (*cf.* définitions 2.1 et 2.2) et de *données de rétroaction* associées à ces ressources (*cf.* section 2.2.2). Dans la configuration la plus simple, une référence peut consister en une simple adresse *URL*, et les données de rétroaction limitées à un effectif qui représente le nombre de personnes ayant apprécié le contenu de la page se trouvant à l'*URL* en question. Une configuration plus complexe consiste à considérer une référence comme étant composée d'un ensemble d'éléments s'ajoutant à l'*URL*, tels qu'un résumé, des *tags*, le nom de l'auteur, la date de publication etc., et des données de rétroaction qui incluent des commentaires, des votes positifs, des votes négatifs, etc. L'ensemble de rétroaction associé à une ressource est représenté par des couples (k, v), où k fait référence à un type de rétroaction (*cf.* section 2.2.2), et v la valeur associée. Les valeurs associées à v peuvent être des votes ou des chaînes de caractères (mot-clés, commentaires, ...). Un couple (k, v) est appelé *objet* de *rétroaction* et l'univers des couples (k, v) est dénoté par Ψ. Le tableau 1 présente quelques notions de base liées à notre étude.

Une ressource accessible via le Web est représentée par les éléments suivants : une adresse url qui permet d'atteindre la ressource, le contenu ct de la ressource (par exemple du texte, des images ou des vidéos), et éventuellement un ensemble d'informations $\mathcal{I}$ (ou métadonnées) décrivant la ressource (*cf.* définition 2.1). L'ensemble de toutes les ressources Web est dénoté par $\mathcal{R}$.

Définition 2.1. Ressource Web

Une ressource Web $i \in \mathcal{R}$ se définit par $\langle url, ct, \mathcal{I} \rangle$, où :
- $url \in URLS_i$;
- $ct = \varpi_i$;
- $\mathcal{I} \subseteq \chi_i$.

En pratique, les ressources Web ne sont pas souvent présentes dans les sites Web qui les citent (comme cela est le cas des sites Web communautaires), et seulement des références à ces ressources sont présentes.
Une *référence à une ressource Web* est composée d'un ensemble d'informations qui permet de représenter une partie ou la totalité du contenu de la ressource en question. Dans la suite, nous nous basons sur la notion de ressource Web pour donner une définition formelle de la notion de *référence à une ressource Web* (*cf.* définition 2.2).

Définition 2.2. Référence à une ressource Web
Soient une ressource i de $\mathcal{R}$ et ρ une relation de l'ensemble des ressources $\mathcal{R}$ vers un ensemble de triplets composés des éléments des univers de tous les URLs, tous les contenus et de toutes les informations qui décrivent/complètent les ressources ($URLS \times \varpi \times \chi$). L'ensemble des références à la ressource i[1] est représenté par $\rho(i) = \{\langle url, ct, \mathcal{I}\rangle | url = i.url \wedge ct \subseteq i.ct \wedge \mathcal{I} \subseteq i.\mathcal{I}\}$.

Par exemple, la figure 3 représente une référence à une ressource localisée par l'URL : $http : //blogs.sun.com/nbprofiler/entry/\ monitoring\_java\_processes\_running\_as$. Cette référence est une entrée du site $http : //www.dzone.com$ et est composée des éléments suivants :

- url = $http : //blogs.sun.com/nbprofiler/entry/\ monitoring\_java\_processes\_running\_as$;
- ct = "*This post describes how Java processes running as a Windows service can be monitored and/or profiled using VisualVM. By default, when you start VisualVM only Java applications started by the same user are listed in the Applications tree. But sometimes you may need to monitor or profile for example Tomcat running as a Windows service. There are several ways to do it using VisualVM*" ;
- $\mathcal{I} = \{$"*NetBeans Profiler : Monitoring Java Processes Running As a Windows Service*", *Jan 22 200913 :50*, 14, 4, *java, tools, windows*$\}$.

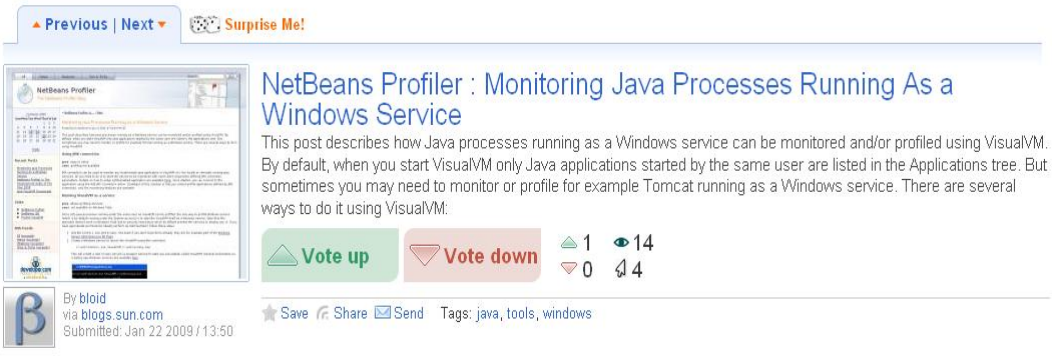

FIG. 3 – *Exemple de référence à une ressource Web.*

[1] La notation $i.url$ représente l'URL de la ressource i.

Les utilisateurs des sites communautaires apportent leurs évaluations sur les ressources dont les références ont été publiées sur le même site. Ces évaluations sont représentées dans ce que nous appelons *instances d'opinion* et se composent de deux parties distinctes. Une première partie sert à identifier la ressource en question, et l'autre partie pour représenter les données de rétroaction associées à cette ressource. De manière formelle, une instance d'opinion se définit comme suit.

Définition 2.3. Instance d'opinion

Une instance d'opinion d'une ressource Web i de $\mathcal{R}$ est dénotée par $o = (\varsigma, \theta)$ où ς est une référence à i, et θ est une liste de données de rétroaction portant sur i. Autrement dit, nous avons :

- $\varsigma \in \rho(i)$;
- $\forall j \in [1..|\theta|] : \theta[j] \in \Psi_i$.

Remarques 2.4.

1. Pour des raisons de concision, un enregistrement de données de rétroaction d'une instance d'opinion $o = (\varsigma, \theta)$ qui se trouve à la position j dans θ sera désigné par $o.\theta[j]$, et la taille de θ sera désignée par $|o.\theta|$. De plus, l'univers de toutes les instances d'opinion sera représenté par $\mathcal{U}_O$.
2. Pour distinguer entre les différentes entrées de $o.\theta$, nous utilisons la notation : $o.\theta^n$ pour faire référence aux entrées dont les valeurs sont numériques et $o.\theta^t$ pour faire référence aux entrées dont les valeurs sont des chaînes de caractères (texte).

Nous venons de voir que les instances d'opinion comportent un certain nombre d'objets de rétroaction. Ces derniers sont classifiés en catégories qu'on appelle aussi classes de rétroaction (*cf.* section 2.2.2).

2.2.2 Langage de description des classes de rétroaction

Syntaxe :

Une classe de rétroaction représente une catégorie d'objets de rétroaction de même type (*cf.* définition ci-dessous). Ces catégories sont directement extraites des sites communautaires utilisés. Par exemple, si un site offre la possibilité de voter pour une ressource, l'échelle des valeurs possibles est explicitement donnée et l'analyse du code HTML du site nous permet de retrouver ce genre d'information de manière semi-automatique.

Définition 2.5. Syntaxe d'une classe de rétroaction

Une classe de rétroaction c, dont l'univers est dénoté par U_Ψ, est définie par le quadruplet $(T, D, Cst, OptimVal)$ où T représente un type de rétroaction, D le domaine de définition de la classe, Cst un ensemble de contraintes sur cette classe de rétroaction, et $OptimVal$ est une valeur optimale pour cette classe.

À titre d'exemple, le quadruplet $(VotePositif, Entiers, \{\geq 0, \leq 10\}, 10)$ est une classe de rétroaction qui représente la catégorie des votes positifs. Chaque vote est un entier qui est supérieur ou égal à 0 et inférieur ou égal à 10. La valeur optimale de cette classe est fixée à 10.

Remarques 2.6.

Dans le cadre de cet article, le calcul de la valeur optimale d'une classe $c \in U_\Psi$ se fait comme suit :

1. Si le domaine de définition de c est numérique alors, $c.OptimVal$ est soit la valeur minimale soit la valeur maximale qui satisfait toutes les contraintes $c.Cst$ dans le domaine de définition de c ($c.D$) ;
2. Si le domaine de définition de c est textuel alors, $c.OptimVal$ est soit l'ensemble vide ($\emptyset$) pour favoriser les textes courts, soit l'infini (∞) pour favoriser plutôt des textes longs.

Sémantique :

Une classe de rétroaction est interprétée par un ensemble d'objets de rétroaction. L'interprétation se base sur deux éléments : (i) le domaine d'interprétation qui est constitué de tous les objets de rétroaction, et (ii) la fonction d'interprétation qui associe à chaque classe un ensemble d'éléments du domaine d'interprétation. De plus, un objet de rétroaction est lié à une classe par la relation d'instanciation qui est définie ci-dessous.

Définition 2.7. Relation d'instanciation

Soit c une classe de rétroaction ($c \in U_\Psi$) et $d = (k, v)$ un objet de rétroaction dans lequel k est un type de rétroaction et v l'instance associée ($d \in \Psi$). On dit que d est instance de c, noté $d \lessdot c$, si et seulement si :

- $d.k = c.T$;
- $d.v \in c.D$;
- $\forall t \in c.Cst, B(t, d.v) = vrai$, c'est-à-dire que $d.v$ satisfait la contrainte t.

Par exemple, l'objet $(VotePositif, 3)$ est une instance de la classe de rétroaction définie par $(VotePositif, Entiers, \{\geq 0, \leq 10\}, 10)$. Par contre, l'objet de rétroaction $(VotePositif, 15)$ n'est pas instance de cette même classe, car la contrainte "≤ 10" n'est pas satisfaite.

Remarque 2.8. Pour désigner la classe d'une instance d'opinion o, la notation $[o]_c$ est utilisée.

Subsomption de classes de rétroaction :

La subsomption de classes de rétroaction est un ordre partiel qui permet d'ordonner un ensemble de classes provenant de différents sites communautaires en hiérarchies. Intuitivement, deux classes de rétroaction sont liées par la relation de subsomption si l'une d'elles se manifeste dans l'autre de manière identique ou avec un domaine de définition plus spécifique et avec un ensemble de contraintes couvrant un domaine plus large. La formalisation de cette relation est donnée dans la définition 2.9 ci-dessous.

Définition 2.9. Subsomption de classes de rétroaction

Soient c_1 et c_2 deux classes de U_Ψ, on dit que c_2 subsume c_1, noté $c_1 \sqsubseteq c_2$, si et seulement si :

1. $c_1.T = c_2.T$;
2. $c_1.OptimVal \leq c_2.OptimVal$;
3. $((\bigcap_{cst_i \in c_1.Cst} D\rfloor_{cst_i}) \cap c_1.D) \subseteq ((\bigcap_{cst_j \in c_2.Cst} D\rfloor_{cst_j}) \cap c_2.D)$, où $D\rfloor_{cst_i}$ et $D\rfloor_{cst_j}$ sont respectivement les restrictions de D par les contraintes cst_i et cst_j.

La dernière condition permet de vérifier que la restriction du domaine de c_1 par l'ensemble de ses contraintes est couvert par la restriction du domaine de c_2 avec les contraintes de ce dernier, car il est possible d'avoir $c_1.D \subseteq c_2.D$ et en même temps d'avoir les contraintes de c_2 plus restrictives que celles de c_1 au point que le domaine effectif de c_1 soit plus étendu que le domaine de c_2.

À titre d'exemple, si $c_2 = (VotePositif, [0..10], \{\}, 10)$, $c_1 = (VotePositif, Entiers, \{\geq 1, \leq 5\}, 10)$, alors $c_1 \sqsubseteq c_2$. Comme on peut le constater, la vérification de la première et de la deuxième conditions de la définition 2.9 est intuitive. Ce qui est par intéressant à considérer est la vérification de la troisième condition de cette définition. En effet, si on s'arête à la vérification des domaines de définition de c_1 et de c_2, on constatera que $c_2.D$ est plus restrictif que c_1 ($[0..10] \subseteq Entiers$). Cependant, l'application des deux contraintes de c_1 ($\{\geq 1, \leq 5\}$) au domaine de ce dernier conduit à l'obtention d'un domaine effectif plus restreint que $[0..10]$ et qui est $[1..5]$. D'où la satisfaction de la troisième condition de la définition de la subsomption de classes de rétroaction.

Il est cependant à noter qu'au sein d'un même site Web, les classes ne sont pas comparables. À titre d'exemple, si une application Web donnée possède une fonctionnalité qui permet de récolter les votes des utilisateurs, un seul système de vote sera utilisé.

Avant d'exposer la méthode que nous proposons pour le calcul de la pertinence d'une ressource Web, nous introduisons la notion d'*entrepôt d'opinions*. L'objectif derrière l'introduction de cette notion est de regrouper les différentes notions présentées dans les sections précédentes (classes de rétroaction, instances d'opinion,...) dans une même entité et pouvoir ainsi identifier et faire la distinction entre les éléments appartenant à différents sites Web communautaires.

2.3 Entrepôt d'opinions

Dans notre modèle de représentation du domaine, un entrepôt d'opinions est défini par un ensemble de classes de rétroaction et un ensemble d'instances d'opinion tel que décrit ci-après.

Définition 2.10. Entrepôt d'opinions

Soient $\mathcal{U}_\Psi$ et $\mathcal{U}_O$ les univers respectifs des classes de rétroaction et des instances d'opinion. Un entrepôt e_i de l'ensemble des entrepôts, désigné par $\mathcal{E}$, est un couple (s_c, s_o) où :

- $s_c \subseteq \mathcal{U}_\Psi$;
- $s_o \subseteq \mathcal{U}_O$;
- $\forall o_i \in s_o, \exists\, c_j \in s_c$ tel que $o_i \lessdot c_j$. Ce qui veut dire que toute instance d'opinion d'un entrepôt est une instance d'une certaine classe de rétroaction du même entrepôt.

La figure 4 schématise les entrepôts d'opinions ainsi que leurs composantes (instances d'opinions et classes de rétroaction).

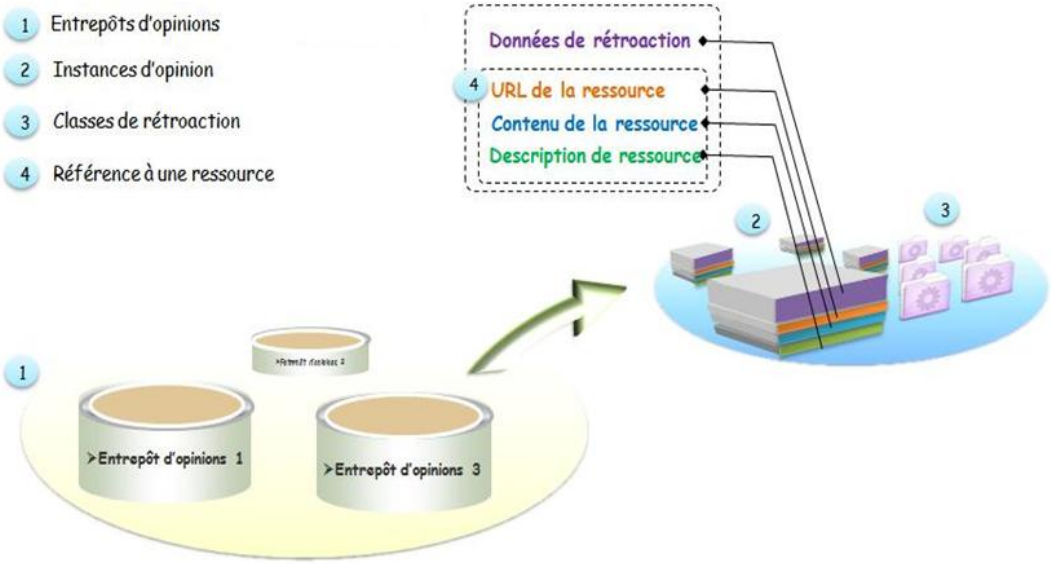

FIG. 4 – *Représentation des données d'opinions par la notion d'entrepôt et de données de rétroaction.*

Pour savoir si un entrepôt est plus général qu'un autre entrepôt, nous introduisons la relation de subsomption d'entrepôts. De même, nous définirons deux instances d'opinion parti-

culières qui seront associées à chaque entrepôt. Il s'agit du *sommet* et du *fond* d'un entrepôt. Ces instances particulières ont un intérêt capital dans le calcul du degré de pertinence des instances d'opinion. En effet, nous avons besoin d'un mécanisme qui nous permet d'ordonner les instances d'opinion afin de pouvoir retourner à l'utilisateur les instances les plus prioritaires seulement.
L'approche naïve consistera à rechercher les instances extrêmes dans chaque entrepôt et de mesurer la distance existante entre chaque instance de l'entrepôt avec ces extrêmes. Cependant, le processus de recherche d'instances extrêmes est coûteux car il nécessite un premier parcours de tout l'entrepôt pour retrouver les extrêmes et un deuxième parcours pour mesurer la distance entre les différentes instances et ces extrêmes.
Le deuxième problème qu'une telle démarche soulève est le caractère variable de ces extrêmes. En effet, une instance qui est extrême à un instant t risque de ne pas l'être à un instant t' (suite à l'ajout de nouvelles instances à l'entrepôt qui vont prendre leur place). Pour maintenir le sytème consistant, après chaque ajout d'instance d'opinion dans un entrepôt, on doit vérifier si les instances déjà sélectionnées gardent toujours leur statut, sinon les remplacer par les nouvelles valeurs extrêmes.
Comme on peut le constater, cette méthode nécessite beaucoup de calcul et exige la surveillance constante des mises à jours des entrepôts. Pour éviter une telle situation, nous proposons de remplacer les instances extrêmes effectives d'un entrepôt par des instances virtuelles. Ces instances sont calculées à partir des valeurs optimales des classes de rétroaction et peuvent ne pas figurer dans l'entrepôt mais elles garantissent la non existence d'autres instances ayant des valeurs plus extrêmes.
L'avantage de ces instances virtuelles est double : (1) leur calcul ne nécessite pas le parcours de l'entrepôt et (2) elles n'induisent pas une vérification continuelle à chaque insertion d'une nouvelle instance dans l'entrepôt, réduisant ainsi la charge en traitement.

2.3.1 Subsomption d'entrepôts d'opinions

La relation de subsomption d'entrepôts d'opinions est un ordre partiel qui permet d'organiser un ensemble d'entrepôts suivant leurs modèles de rétroaction. Cette relation se réduit à l'inclusion ensembliste des classes de rétroaction. Formellement, nous avons ce qui suit.

Définition 2.11. Subsomption d'entrepôts
Soient e_1 et e_2 deux éléments de l'ensemble des entrepôts $\mathcal{E}$, on dit que e_1 est plus spécifique que e_2, ou encore e_1 est subsumé par e_2, noté $e_1 \sqsubseteq e_2$, si et seulement si $\forall c_i \in e_1.s_c, \exists c_j \in e_2.s_c$ tel que $c_i \sqsubseteq c_j$.

2.3.2 Sommet et fond d'un entrepôt

Le sommet d'un entrepôt est une instance d'opinion à laquelle sont associées les valeurs optimales ($OptimVal$) des classes correspondantes. Elle est définie comme suit.

Définition 2.12. Sommet d'un entrepôt d'opinions

Soient e un élément de $\mathcal{E}$ et $o \in \mathcal{U}_O$, l'instance o est dite sommet de e, notée $\widehat{e}$, si et seulement si :

1. $\forall k \in e.s_c : \exists\, j \in o.\theta$ tel que $[j]_c = k$;
2. $\forall j \in o.\theta : \exists\, k \in e.s_o$ tel que $[j]_c = k$;
3. $\forall j \in o.\theta : j.v = [j]_c.OptimVal$.

De la même manière, le fond d'un entrepôt est une instance d'opinion à laquelle sont associées les valeurs qui sont duales aux valeurs optimales des classes de l'entrepôt. Soient x, y deux valeurs d'un domaine numérique D, on dit que y est duale à x si et seulement si :

- $x = min(D)$ et $y = max(D)$;
- $x = max(D)$ et $y = min(D)$.

Dans le cas où x et y sont des chaînes de caractères (ensemble de mots), on dit que y est duale à x si et seulement si :

- $x = \emptyset$ et $y = \infty$;
- $x = \infty$ et $y = \emptyset$.

Définition 2.13. Fond d'un entrepôt d'opinions

Soient e un élément de $\mathcal{E}$ et $o \in \mathcal{U}_O$, l'instance o est dite fond de e, notée $\underline{e}$, si et seulement si :

1. $\forall k \in e.s_c : \exists\, j \in o.\theta$ tel que $[j]_c = k$;
2. $\forall j \in o.\theta : \exists\, k \in e.s_o$ tel que $[j]_c = k$;
3. $\forall j \in o.\theta^n : j.v \in \{Max([j]_c.D \cup (\bigcap_{cst_k \in ([j]_c.Cst)} D\rfloor_{cst_k})), Min([j]_c.D \cup (\bigcap_{cst_k \in ([j]_c.Cst)} D\rfloor_{cst_k}))\} - \{[j]_c.OptimVal\}$;
4. $\forall j \in o.\theta^t : j.v \in \{\emptyset, \infty\} - \{[j]_c.OptimVal\}$.

Les deux premières conditions permettent de vérifier que pour toute classe de rétroaction d'un entrepôt donné est associée une seule instance de données de rétroaction du fond de ce même entrepôt. La troisième condition permet de vérifier que chaque donnée de rétroaction de valeur numérique est la valeur duale à la valeur optimale : si la valeur optimale est le maximum des valeurs possibles, alors la donnée de rétroaction dans le fond sera la valeur minimale et vis versa. La dernière condition est similaire à la troisième condition sauf qu'elle s'occupe de vérifier que les données de rétroaction de valeurs textuelles représentent exactement la valeur duale de la valeur optimale : si la valeur optimale est l'infini, alors la donnée de rétroaction dans le fond sera l'ensemble vide et vis versa.

2.3.3 Distance entre deux instances d'opinion

Comme nous l'avons déjà indiqué, les composantes d'une instance d'opinion peuvent être soit des nombres (nombre de personnes qui apprécient/n'apprécient pas la ressource, nombre

de commentaires, ...) soit des chaînes de caractères (résumé, les mots clés, titre, ...). Dans cette section, nous présentons une formule de pertinence qui prend en considération les données textuelles et numériques des instances d'opinion. Pour mesurer la distance entre les composantes numériques de deux instances d'opinion dans un entrepôt donné, nous utilisons la formule de *Minkowski* d'ordre p (Varadhan et Manocha, 2006) donnée ci-dessous. L'ordre p représente le nombre de classes de rétroaction que comporte chaque entrepôt.
Par contre, le calcul de la distance existante entre les composantes textuelles des instances d'opinion se base sur la proportion des mots qui sont différents dans les deux textes. Il est à noter que d'autres mesures pour le calcul de la distance entre deux chaînes de caractères existent. Nous citons en particulier la distance de *Levenshtein* (Runkler et Bezdek, 2007; Yujian et Bo, 2007) où la distance entre deux mots est donnée par le nombre minimal de remplacements, ajouts et suppressions de lettres pour passer d'un mot à un autre.

La formule globale de mesure de la similarité de deux instances d'opinion est décrite dans la définition 2.14

Définition 2.14. Distance entre deux instances d'opinion

Soient $e \in \mathcal{E}$ et o_1, o_2 deux instances d'opinion de l'entrepôt e ($o_1 \in e.s_o$ et $o_2 \in e.s_o$). La distance entre o_1 et o_2, notée $d_e(o_1, o_2)$, est comme suit : $d_e(o_1, o_2) = d_e^n(o_1, o_2) + d_e^t(o_1, o_2)$ où :

– $d_e^n(o_1, o_2)$ est la distance de *Minkowski* entre les composantes numériques de o_1 et o_2 qui est donnée par la formule suivante :

$$d_e^n(o_1, o_2) = \sqrt[|e.s_c|]{\sum_{j \in [1..|o_1.\theta^n|]} (o_1.\theta^n[j].v - o_2.\theta^n[j].v)^{|e.s_c|}}$$

– $d_e^t(o_1, o_2)$ est la distance entre les composantes textuelles de o_1 et o_2 tel que :

$$d_e^t(o_1, o_2) = \begin{cases} 0, & |\alpha| \in 0, \infty \text{ et } |\beta| = |\alpha| \text{ ;} \\ 1, & (|\alpha| \in 0, \infty \text{ et } |\beta| \neq |\alpha|) \text{ ou} \\ & (|\beta| \in 0, \infty \text{ et } |\beta| \neq |\alpha|) \text{ ;} \\ \frac{\frac{|\alpha - \beta|}{|\alpha|} + \frac{|\beta - \alpha|}{|\beta|}}{2}, & \text{sinon.} \end{cases}$$

Avec $\alpha = \bigcup_{j \in [1..|\theta|]} o_1.\theta^t[j].v \cup o_1.\varsigma.ct \cup o_1.\varsigma.\mathcal{I}$, $\beta = \bigcup_{j \in [1..|\theta|]} o_2.\theta^t[j].v \cup o_2.\varsigma.ct \cup o_2.\varsigma.\mathcal{I}$ et $|texte|$ représente le nombre de mots contenus dans le texte $texte$.

2.4 Mesure de la pertinence d'une instance d'opinion

La pertinence d'une instance d'opinion o de l'entrepôt e donnée par $\mathcal{P}_e(o)$, est la distance qui sépare o du fond de e, c'est-à-dire $\mathcal{P}_e(o) = d_e(o, \underline{e})$.

Les valeurs retournées par $\mathcal{P}_e$ ne sont pas normalisées. Or, pour pouvoir classer deux objets appartenant à des entrepôts différents, nous avons besoin d'une mesure normalisée. Pour cela, nous proposons la fonction $\overline{\mathcal{P}_e}$ qui calcule une valeur de pertinence normalisée dans l'intervalle $[0..1]$. La fonction $\overline{\mathcal{P}_e}$ est définie comme suit :

Définition 2.15. Pertinence normalisée
Soient $e \in \mathcal{E}$, $o \in e$, et $x = \mathcal{P}_e(o)$. La pertinence normalisée de o dans e est donnée par $\overline{\mathcal{P}_e}(x)$ tel que $\overline{\mathcal{P}_e}$ est définie de $\mathcal{U}_O$ vers $[0..1]$ comme suit :

$$\overline{\mathcal{P}_e}(o) = \begin{cases} 0, & \text{si} \mathcal{P}_e(\widehat{e}) = 0; \\ |\frac{\mathcal{P}_e(o)}{\mathcal{P}_e(\widehat{e})}|, & \text{sinon.} \end{cases}$$

Il est à remarquer que la pertinence maximale d'une instance d'opinion est atteinte lorsque tous les objets de rétroaction qui lui sont associés ont des valeurs égales aux valeurs optimales des classes de rétroaction correspondantes.

Maintenant que la fonction $\overline{\mathcal{P}_e}$ permet de calculer des valeurs de pertinence normalisées, les ressources décrites par les instances d'opinion de différents entrepôts peuvent être classées.

3 Interrogation d'un ensemble d'entrepôts d'opinions

À chaque fois qu'un utilisateur soumet une requête, les mots clés utilisés sont récupérés sans lemmatisation mais en supprimant les termes usel (exemple : le, les,la,...) et le système interroge la base d'entrepôts. Le résultat de l'interrogation des entrepôts est une collection d'instances d'opinion. Ces instances sont ainsi classées par ordre décroissant des valeurs de $\overline{\mathcal{P}_e}$ et un maximum de k premiers éléments sont renvoyés à l'utilisateur (*cf.* définition 3.1 ci-dessous).

Définition 3.1. K-résultats Soient $\mathcal{E}_1$ un ensemble d'entrepôts ($\mathcal{E}_1 \subseteq \mathcal{E}$), Q un ensemble de mots clés et k un entier positif. Le résultat d'interrogation de $\mathcal{E}_1$ par le biais de Q est une collection d'instances d'opinion éléments de $\mathcal{E}_1$. Ces instances sont données par la fonction $\mathcal{F}_{\mathcal{E}_1}$, tel qu'une instance d'opinion o_i appartient à $\mathcal{F}_{\mathcal{E}_1}(Q, k)$ si et seulement si :

1. $\exists\, e \in \mathcal{E}_1$ tel que $o_i \in e$;
2. $\exists\, t \in Q$ tel que $t \in (o_i.\varsigma.ct \cup o_i.\varsigma.\mathcal{I})$;
3. $k' \leq k$ où $k' = |\{o_j \in U_o | (\exists\, e \in \mathcal{E}_1$ tel que $o_j \in e) \wedge (\exists\, t \in Q$ tel que $t \in (o_j.\varsigma.ct \cup o_j.\varsigma.\mathcal{I})) \wedge (\frac{\sum_{e \in \mathcal{E}_1} \overline{\mathcal{P}_e}(o_j)}{|\mathcal{E}_1|} > \frac{\sum_{e \in \mathcal{E}_1} \overline{\mathcal{P}_e}(o_i)}{|\mathcal{E}_1|})\}|$.

Les deux premières conditions de la définition 3.1 garantissent respectivement l'appartenance de l'instance o_i à un entrepôt de $\mathcal{E}_1$ et de l'existence d'au moins un mot clé de la requête qui figure dans la description du contenu de o_i. La troisième condition quant à elle, vérifie que o_i est parmi les k instances les plus pertinentes en calculant le nombre d'instances qui sont plus pertinentes que l'instance courante. Il est à noter que les valeurs de pertinence des instances d'opinion sont calculées une seule fois et éventuellement mises à jour lors des modifications des profils de ces instances.

4 Évaluation de l'apport du cadre proposé

L'objectif de notre étude de l'apport du cadre de recherche web basée sur les données d'opinion que nous proposons est d'évaluer la pertinence des résultats qu'on peut obtenir avec

ce système. Pour atteindre cet objectif, nous avons procédé en deux phases. Dans la première phase, nous avons implémenté un prototype expérimental du modèle théorique présenté dans les sections 2 et 3 (*cf.* sous-section 4.1). Par la suite, nous avons évalué le système par rapport à quelques systèmes existants.

Concernant cette deuxième phase, il nous a semblé plus raisonnable d'effectuer une évaluation empirique de notre approche. Autrement dit, cela consiste à comparer les évaluations des utilisateurs associées aux résultats retournés par notre approche qui est basée sur les données d'opinions avec les résultats d'un moteur de recherche d'information, dit classique, et qui est *Google*. Comme la pertinence d'un résultat dépend d'une évaluation humaine, nous avons décidé de faire appel à des sujets externes dont le rôle est de soumettre des requêtes de recherche et d'indiquer le degré de pertinence des résultats (*cf.* 4.1.1).

4.1 Implémentation

SocialSeeker est le nom du moteur de recherche prototype que nous avons implémenté. L'objectif premier de son développement est de montrer la faisabilité de l'approche que nous avons proposée. Ce moteur a été implémenté en utilisant *Rails 2.02* (Hansson, 2008), un cadre pour le développement d'applications Web qui se base sur le paradigme *Modèle-Vue-Contrôleur* (*MVC*) (Gamma et al., 1995). Ainsi, le code est principalement écrit en utilisant *Ruby* (Ruby, 2008), un langage de programmation dynamique, et HTML/CSS pour la programmation des interfaces. Le moteur est déployé en phase de tests sur un serveur d'applications *Thin* (Cournoyer, 2008) et *Nginx* (Sysoev, 2008) en tant que serveur frontal. Actuellement, *SocialSeeker* exploite deux entrepôts d'opinions : (*Dzone*) et (*Reddit*). Pour scruter ces applications Web, nous avons utilisé la librairie *hpricot* (Why, 2008) écrite en *Ruby*.

4.1.1 Protocole d'expérimentation et évaluation empirique de l'approche

Pour mener les expérimentations avec un maximum d'objectivité, nous avons élaboré un protocole. Les différentes étapes de ce protocole sont comme suit :

1. Mettre à la disposition de l'utilisateur une interface unifiée pour qu'il puisse effectuer une recherche sur le web via une requête composée de mots clés ;
2. Récupérer la requête de recherche d'un utilisateur ;
3. Exécuter cette requête sur *SocialSeeker* ;
4. Exécuter la même requête sur *Google* ;
5. Assembler les cinq premiers résultats dans une même liste (10 éléments) en alternant entre les résultats des deux moteurs ;
6. Retourner les résultats à l'utilisateur sans lui indiquer le système qui les a retournés ;
7. Demander à l'utilisateur d'indiquer le(s) résultat(s) qui est (sont) pertinent(s) à sa recherche ;
8. Récupérer et analyser les données retournées par les utilisateurs.

L'analyse des données retournées par les utilisateurs est une tâche relativement simple. Cela consiste à mesurer le taux de satisfaction des utilisateurs par rapport aux résultats donnés

par les deux moteurs de recherche. Autrement dit, dans notre expérimentation, la pertinence des résultats d'une requête est mesurée par le degré de satisfaction de l'utilisateur qui a soumis la requête.

C'est ainsi que nous avons mené une évaluation empirique de notre approche avec neuf sujets humains. Étant donné que l'un des entrepôts d'opinions utilisé se focalise sur le domaine des nouvelles technologies de l'information, et pour ne pas biaiser les recherches avec l'autre moteur de recherche considéré, qui est plutôt un moteur de recherche "généralisé", nous avons proposé aux utilisateurs de faire des recherches dans le domaine des nouvelles technologies. Chaque utilisateur a effectué un certain nombre de requêtes et a évalué les éléments retournés par les moteurs de recherche utilisés (*SocialSeeker* et *Google*). Il est à noter que le système ajoute à chaque élément du résultat retourné deux options : apprécie et n'apprécie pas. Lors de l'évaluation, l'utilisateur a juste à choisir parmi ces deux options.

Les chiffres liés à l'activité des différents utilisateurs sont présentés dans le tableau 2. Alors que la première ligne présente le nombre de requêtes effectuées par chaque utilisateur, la deuxième ligne présente le nombre d'éléments de résultats qui ont été évalués (positivement ou négativement) par les utilisateurs.

	U_1	U_2	U_3	U_4	U_5
Requêtes	15	8	12	30	16
Résultats évalués	3	22	16	72	44
En prevenance de *SocialSeeker*	2	13	4	10	37
En prevenance de *Google*	1	9	12	62	7
	U_6	U_7	U_8	U_9	
Requêtes	7	24	12	4	
Résultats évalués	9	69	43	15	
En provenance de *SocialSeeker*	5	24	20	14	
En provenance de *Google*	4	35	23	1	

TAB. 2 – *Nombre de requêtes et d'éléments résultant qui sont évalués par chaque utilisateur.*

À partir du tableau 2, nous avons calculé la proportion des évaluations positives pour chaque utilisateur et pour respectivement chacun des deux moteurs de recherche utilisés. Ces données sont présentées dans le tableau 3.

Le tableau 4 présente deux mesures liées aux données du tableau 3. La première colonne de ce tableau présente la proportion moyenne des évaluations positives selon les utilisateurs et suivant les résultats retournés par un moteur de recherche donné. Nous avons aussi calculé la proportion moyenne des évaluations positives selon le nombre d'évaluations des différents utilisateurs (*cf.* colonne 2 du tableau 4).

	U_1	U_2	U_3	U_4	U_5
SocialSeeker					
Évaluation positive	100%	76,92%	75%	90%	89%
Google					
Évaluation positive	100%	100%	33,33%	22,58%	42,85%
	U_6	U_7	U_8	U_9	
SocialSeeker					
Évaluation positive	100%	29%	45%	71,42%	
Google					
Évaluation positive	75%	91,42%	95%	100%	

TAB. 3 – *Proportion des résultats évalués par les différents utilisateurs par moteur de recherche.*

	Proportion moyenne des évaluations positives selon les utilisateurs	Proportion moyenne des évaluations positives selon l'ensemble des évaluations
SocialSeeker	75,14%	68,21%
Google	73,35%	57,79%

TAB. 4 – *Proportion des résultats évalués positivement par utilisateur et par moteur de recherche.*

Analyse des résultats

Nous expliquons les écarts dans le tableau 3 principalement par le type d'informations recherchées. Lorsque les utilisateurs recherchent des informations techniques et d'actualité dont tout le monde parle (effet de mode), *SocialSeeker* semble données de meilleurs résultats. En effet, les données utilisées dans *SocialSeeker* reflètent directement les préoccupations/intrérêts tels que formulés sur les sites communautaires. Par contre, lorsque les utilisateurs recherches des informations moins techniques ou bien des données corporatives, *Google* a tendance à donner de meilleures résultats.

Tel que montré par le tableau 4, les proportions moyennes des évaluations positives selon les utilisateurs et selon l'ensemble des évaluations de notre système sont légèrement supérieures à celles calculées pour le moteur de recherche *Google*. L'une des justifications de cette amélioration des résultats de recherche est liée à la source même d'information utilisée (sites communautaires spécialisés) et à la notoriété de la source originale. À titre d'exemple, en effectuant une recherche avec le mot clé *java*, les premiers éléments de résultat retournés par *Google* sont des liens vers les sites de *Sun* (Sun, 2008) ou d'entités affiliées, alors qu'avec *SocialSeeker* on retrouve des liens à des articles et des ressources qui sont d'actualité (critique d'un livre, tendances, articles spécialisés, entrées de blogs, ...).
Le premier motif qui se dégage est le suivant : dans les moteurs de recherche classiques, la pertinence calculée des pages Web semble être biaisée par la grandeur/renommée des sites de ces pages. Le deuxième motif est que les internautes sont plutôt intéressés par du contenu qui

n'est pas forcément publié sur les sites des grandes compagnies du monde des technologies de l'information et par un contenu que leur pairs ont évalué positivement.

La lecture des résultats de ces premières expérimentations est encourageante. Cependant, pour tirer des conclusions définitives quant à la pertinence des recherches avec *SocialSeeker*, des expériences à grandes échelle sont nécessaires. Ces dernières devraient porter sur une large base d'utilisateurs et concerner un plus large spectre des communautés Web couvertes par le moteur. La prochaine étape pour aboutir à cet objectif consiste à faire passer *SocialSeeker* du mode prototype de test au mode production. Cela va non seulement nous permettre de confirmer/infirmer les résultats obtenus par ces premières expérimentations mais aussi nous permettre de tester la mise à l'échelle du modèle proposé.

5 Conclusion et perspectives

Dans cet article, nous avons proposé un cadre d'application pour l'intégration des données d'opinion dans les moteurs de recherche. Il est principalement composé d'un langage de description des données et d'un ensemble d'opérations sur les éléments de ce langage. Ce cadre tient compte de l'hétérogénéité des données et permet de comparer des données d'opinion en provenance de différentes sources par l'utilisation d'une mesure normalisée pour le calcul de la pertinence. Cette dernière prend en considération les évaluations positives et les évaluations négatives des internautes.

Le modèle proposé est ensuite utilisé pour le développement de *SocialSeeker*, un prototype de moteur de recherche qui exploite les données d'opinion disponibles sur le Web pour le classement des ressources. Les résultats des expérimentations que nous avons menées sont assez encourageants et indiquent que les données d'opinions peuvent être exploitées pour une recherche d'informations sur le Web plus efficace que les moteurs de recherche classiques.
Il est cependant à noter que l'intégration des données d'opinion n'est pas sans risque. En effet, il est probable que le calcul de la pertinence soit biaisé par différents facteurs, tels que le degré d'expertise des utilisateurs/évaluateurs, de leur subjectivité et du contexte général dans lequel les évaluations ont été récoltées et des mesures de distances utilisées. Concernant ce dernier point, l'application d'une méthode d'apprentissage de métrique (Lebanon, 2006) devrait améliorer davantage la qualité des résultats de recherches d'information. En effet, cette technique permettra la sélection de métrique adéquate suivant le contexte d'utilisation.

En outre, l'implémentation actuelle est au stade prototype et son extension permettra de réaliser des expérimentations à grande échelle. L'une des limites de l'implémentation actuelle est liée à la méthode d'enregistrement des nouveaux entrepôts d'opinions qui pour l'instant se fait manuellement. Une extension possible serait de développer autour de *SocialSeeker* une *API* (*Application Programming Interface*) qui facilitera la gestion des entrepôts en tant que services Web de manière programmable. De plus, il serait intéressant de comparer les résultats de l'utilisation de différentes formules de calcul de distance entre deux instances d'opinion.
Dans cette nouvelle version de *SocialSeeker*, les utilisateurs auront la possibilité de participer à l'enrichissement des entrepôts d'opinions en ajoutant des données de rétroaction directement à partir de l'interface de *SocialSeeker*. Finalement, nous pensons que les moteurs de recherche classiques pourraient bénéficier de ce travail pour améliorer la pertinence de leurs résultats en intégrant les données d'opinion dans le processus de classification des pages Web.

Références

Adah, S., C. Bufi, et Y. Temtanapat (1997). Integrated search engine. *KDEX '97 : Proceedings of the 1997 IEEE Knowledge and Data Engineering Exchange Workshop*, 140–147.

Adda, M. (2008). Socialseeker : community-based search engine. *http ://socialseeker.no-ip.org/*.

Adda, M., R. Missaoui, et P. Valtchev (2008). Vers la recherche d'informations sur le web basée sur des données d'opinion. *Atelier FOuille des Données d'OPinions (FODOP'08), conjointement à la conférence INFORSID*.

Baader, F., D. Calvanese, D. McGuinness, D. Nardi, et P. Patel-Schneider (2003). The description logic handbook : Theory, implementation and applications. *Cambridge University Press*, 574.

Bojrs, U., J. G. Breslin, A. Finn, et S. Decker (2008). Using the semantic web for linking and reusing data across web 2.0 communities. *Web Semantics : Science, Services and Agents on the World Wide Web 6*(1), 21–28.

Borgatti, S.-P. et R. Cross (2003). A relational view of information seeking and learning in social networks. *Manage. Sci. 49*(4), 432–445.

ChaCha (2008). Chacha - mobile search | text search | questions and answers. *http ://www.chacha.com*.

Cournoyer, M.-A. (2008). Thin : open-source, ruby web server. *http ://code.macournoyer.com/thin/*.

Digg (2008). Digg - all news, videos, & images. *http ://www.digg.com*.

Ding, C. et C.-H. Chi (2003). A generalized site ranking model for web ir. *Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, 584–587.

Dipiti (2008). Dipiti. funny name for human-filtered search. *http ://www.dipiti.com*.

Dzone (2008). Dzone.com - fresh links for developers. *http ://www.dzone.com*.

Gamma, E., R. Helm, R. Johnson, et J. Vlissides (1995). *Design Patterns*. Addison-Wesley Professional.

Google (2008). Google. *http ://www.google.com*.

Hansson, D. H. (2008). Ruby on rails : open-source web framework. *http ://www.rubyonrails.org*.

Hooff, B., W. Elving, J. Meeuwsen, et C. Dumoulin (2003). Knowledge sharing in knowledge communities. pp. 119–141.

Joachims, T., L. Granka, B. Pan, H. Hembrooke, F. Radlinski, et G. Gay (2007). Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Trans. Inf. Syst. 25*(2), 7.

Lebanon, G. (2006). *Metric learning for text documents*, Volume 28.

Lewis, D. (2006). What is web 2.0 ? *ACM Crossroads journal 13*(1), 3–3.

Mahalo (2008). Mahalo.com : Human-powered search. *http ://www.mahalo.com*.

Microsoft (2008). Microsoft live search. *http ://www.microsoft.com*.

Reddit (2008). Reddit.com : what's new online ! *http ://www.reddit.com*.

Ruby (2008). Ruby programming language. *http ://www.ruby-lang.org/en*.

Runkler, T. et J. Bezdek (2007). *Automatic keyword extraction with relational clustering and Levenshtein distances*, Volume 2.

Shahabi, C. et Y.-S. Chen (2003). An adaptive recommendation system without explicit acquisition of user relevance feedback. *Distrib. Parallel Databases 14*(2), 173–192.

Slashdot (2008). Slashdot - news for nerds, stuff that matters. *http ://www.slashdot.org*.

Sun (2008). Sun microssystems, inc. *http ://java.sun.com/*.

Sun, J., H. Zeng, H. Liu, Y. Lu, et Z. Chen (2005). Cubesvd : A novel approach to personalized web search. *14th international conference on World Wide Web*, 574.

Sysoev, I. (2008). Nginx : open-source, high-performance http server and reverse proxy. *http ://wiki.codemongers.com/*.

Varadhan, G. et D. Manocha (2006). *Accurate Minkowski sum approximation of polyhedral models*, Volume 68. San Diego, CA, USA : Academic Press Professional, Inc.

Why (2008). Hpricot : flexible html/xml parser. *http ://code.whytheluckystiff.net/hpricot/*.

Yahoo (2008). Yahoo ! *http ://www.yahoo.com*.

Yujian, L. et L. Bo (2007). *A Normalized Levenshtein Distance Metric*, Volume 29.

Summary

Web Search Engines aim to assist users in finding relevant information. To measure the relevance of a Web page, different strategies are used. However, page ranking is mainly conducted by relying on automatic assessment criteria. Hence, a gap is created between the effective relevance of a content and the computed one.
To reduce this gap, the present paper introduces a framework for feedback-based research engine development. To illustrate the effectiveness and use of the proposed framework, we developed a web search engine called *SocialSeeker*. Finally, a preliminary evaluation of the proposed prototype is presented and discussed.