Recherche adaptative de documents textuels

Rachid Arezki

3 octobre 2005

Table des matières

Chapitre 1

Introduction

Avec le développement d'Internet et des supports de stockage, la quantité de documents disponibles ne cesse de croître. Il est donc nécessaire de disposer de systèmes capable d'appréhender de manière plus efficace ces quantités énormes de documents.

Les systèmes classiques de recherche d'information sont d'une grande utilité et permettent de retrouver de l'information à travers des requêtes souvent exprimés en langage naturelle, et retourne à l'utilisateur une liste de documents ordonnées selon leur pertinence. Cependant de nombreux problèmes persistent pour permettre à l'utilisateur d'obtenir les ressources souhaitées d'une manière efficace.

de récentes études empiriques effectuées montrent que les requêtes de recherche des usagers sont pauvrement formulés et il ne visualisent pas plus de deux documents.

D'une part la difficulté pour les utilisateur de formuler leur requête de recherche de manière efficace. En effet il a été montré que la faible pertinence des systèmes de recherche est du principalement a des requêtes mal formés.

D'autre part, une même requête utilisateur peut correspondre à différents besoins. En effet à partir d'une même requête un document apparaîtra plus ou moins pertinent par rapport au point de vue et au connaissances de tel ou tel utilisateur.

Une solution possible consiste d'adapter les systèmes de recherche d'information au profils des utilisateurs afin d'optimiser la recherche d'information.

En autre les systèmes de recherche d'information requiert la formulation de requêtes à chaque nouveau besoins d'information.

Ainsi, notre travail se situe dans le contexte de l'accès intelligent à l'information et plus particulièrement l'accès personnalisé à l'information. Nous nous limitons

dans cette thèse aux informations textuelles.

dire quelque soi la méthode de personnalisation on a besoins de représenter les centres d'intérêts de l'utilisateur sous forme de profils.

bla bla bla

1.1 Problématique

? ? les limites des approches

- comment modéliser les intérêts de l'utilisateur.
- comment sélectionner à partir du profil l'information pertinente.

bla bla bla

1.1.1 recherche personnalisé d'information

1.1.2 recommandation d'informations

1.2 Contributions

Notre travail rentre dans la catégorie des systèmes de personnalisation de l'information plus particulièrement les systèmes de recherche personnalisé de l'information et les systèmes de recommandation de l'information. Notre contribution se situe à plusieurs niveaux et concernent plus précisément l'apprentissage des profils et leurs intégration dans le processus de personnlisation, ainsi :

1. Concernant la recherche personnalisé de l'information : nous le modèle PDIIR, une approche de recherche d'information basée sur le profil utilisateur. L'originalité de notre approche est l'intégration de la connaissance sur l'utilisateur (le profil utilisateur), dans le processus d'indexation des documents, afin d'adapter la recherche au profil de l'utilisateur.

2. Concernant la recommandation de documents : nous proposons une approche approche pour l'apprentissage des intérêts long terme de l'utilisateur pour la recommandation de documents textuels. Cette approche est basée sur l'analyse de l'évolution dans le temps des classes de documents consultés par l'utilisateur. Le but de notre approche est de déterminer le mieux possible les classes d'intérêts de l'utilisateur afin de donner plus d'importance dans le processus de

recommandation au classes de documents régulièrement consultés qu'à celles concentrées sur de courtes période.

1.3 Plan du mémoire

L'objectif des chapitres 1 et 2 est de présenter les concepts clés liés à notre contribution, à savoir : la recherche et la personnalisation de l'information.

Ainsi, au chapitre 1, nous dressons un état de l'art sur les systèmes de recherche d'information. Nous commencerons par la description des concepts de base de la recherche d'information, suivi des méthodes de sélection et de pondération des termes d'indexation. Ensuite nous décrirons les différents modèles de recherche d'information suivi des méthodes de reformulation de requêtes.

Au chapitre 2, nous présentons les systèmes de personnalisation d'information : Après une introduction sur la notion de profil utilisateur, nous verrons les différentes méthodes de modélisation du profil. Nous présentons ensuite différents catégories de systèmes de personnalisation de l'information, une attention particulière sera consacré au systèmes de recherche d'information personnalisé ainsi qu'au systèmes de collecte passive d'information.

Les chapitres 3, 4 sont consacrés à notre contribution :

Ainsi, nous présentons dans le chapitre 4, le modèle PDIIR, une approche pour la recherche personnalisée d'information. Ainsi, après une description et une justification du modèle nous présentons une série d'expérimentation effectués sur des corpus de références. nous présentons ensuite l'agent PAWebSearch un agent personnel pour la recherche d'information sur le WEb utilisant l'approche PDIIR. Enfin, Enfin nous terminons par une discussion sur les avantages et les limites de notre approche

Dans le chapitre 5, nous présentons une approche pour l'apprentissage des intérêts long terme de l'utilisateur pour la recommandation de documents textuels. Ainsi, après une présentation des problématiques de la modélisation des intérêts long terme de l'utilisateur pour la recommandation de documents, nous présentons l'architecture fonctionnelle de notre système de recommandation LUCI. Nous décrivons en-

suite de manière détaillée notre approche pour la modélisation de l'intérêt long terme de l'utilisateur. nous présentons ensuite une série d'expériences sur un corpus documentaire de notre approche. Enfin nous terminons par une discussion sur les avantages et les limites de notre approche

Enfin en conclusion nous dressons un bilan de nos travaux et présentons les perspectives associés.

Chapitre 2

Les systèmes de recherche d'information

Sommaire

2.1 Introduction

La recherche d'information (RI) peut se définir comme l'ensemble des opérations effectuées pour retrouver des informations répondant à un besoin d'information. Un besoin d'information peut prendre différentes formes : recherche de documents, recherche des références de l'ouvrage d'un auteur, recherche sur un sujet, recherche directe d'un texte, etc.

Dans le cadre de ce mémoire nous nous intéressons plus particulièrement à la recherche documentaire. Celle-ci consiste à trouver dans une collection de documents les granules de documents susceptibles de répondre au besoin d'information de l'utilisateur. Un granule de document correspond à tout ou une partie du document. Il représente l'unité sélectionnée en réponse à une requête utilisateur. Nous nous limitons dans notre étude aux granules de documents textuels. Dans le reste de ce rapport, nous utilisons indifféremment les termes information ou document pour désigner un granule de document.

Dans ce chapitre, nous abordons les systèmes de recherche d'information (SRI) d'une manière générale. Ainsi, dans la section 2.2 nous abordons brièvement les éléments de base de la recherche d'information, tel que la collection de documents, le besoin d'information, la représentation des documents, etc. Dans la section 2.3 nous aborderons l'élément principale de la RI à savoir la représentation des documents, ou nous nous intéresserons plus particulièrement aux méthodes de choix et de pondération des termes d'indexation. Dans la section 2.4, nous présentons les différentes classes de modèles de recherche d'information, où pour chaque classe nous décrivons les systèmes les plus significatifs. Nous nous intéressons dans la section suivante aux méthodes d'expansion de requêtes, ainsi après une présentation des différentes classes d'approches existantes. Nous nous intéressons plus particulièrement aux approches d'expansion automatique de requêtes, où nous décrivons les techniques les plus significatives. Nous présentons dans la section 2.6 les techniques d'évaluation des SRI.

2.2 Éléments de base de la recherche d'information

L'objectif d'un système de recherche d'information est de trouver parmi un ensemble de documents ceux qui sont susceptibles de répondre à un besoins d'informa-

tion de l'utilisateur. Ainsi comme le montre la figure 2.1, un système de recherche d'information est composé des éléments de base suivants :

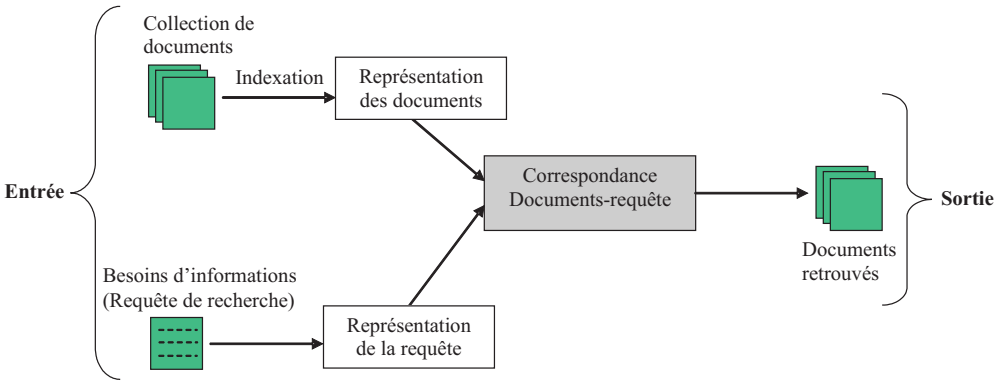

FIG. 2.1 – Architecture de base d'un système de recherche d'information

1. **Une collection de documents** : La collection de documents représente l'ensemble des informations exploitables, compréhensibles et accessibles par l'utilisateur. Une collection comporte un ensemble de granules documentaires.
2. **Une requête de recherche** :
   C'est une expression d'un besoin d'informations de l'utilisateur. Il existe divers type de langages d'interrogation, ainsi une requête peut être exprimé en langage naturel (exemple, "document sur la recherche d'information"), en langage booléen (exemple, "recherche ET Information), à partir d'une interface graphique (formulaires, navigation, etc.).
3. **Représentation des documents et des besoins en information de l'utilisateur** :
   Avant de pouvoir effectuer la recherche proprement dite, les documents et la requête sont souvent transformés d'une description brut (souvent du texte) à une description plus structurée. Cette dernière peut correspondre à des vecteurs [Luh58, Sal71], un réseaux de neurones [Moz99, Kwo89], etc (voir les modèles de RI en section 3.4). Cette représentation nécessite au préalable la définition d'un langage de représentation des documents (appelé aussi termes d'indexation), c'est à dire l'ensemble des concepts susceptible de représenter les documents.
4. **Correspondance documents-requête** : Elle correspond à la recherche proprement dite, c-à-d la mise en correspondance des représentations sémantiques

des documents et d'une représentation sémantique de la requête. Par exemple dans le cadre du modèle vectoriel, l'appariement documents-requête correspond au calcul de similarité entre le vecteur représentant la requête et chacun des vecteurs documents de la collection.

2.3 Les termes d'indexation

Les termes d'indexation désignent l'ensemble des concepts susceptibles de représenter les documents. Les concepts peuvent être des termes simples ou des groupes de termes. Le processus du choix du langage de représentation des documents est appelé *indexation*.

L'indexation peut être manuelle, réalisée par un expert, ou automatique. Le choix de l'une ou de l'autre dépend de plusieurs paramètres dont le plus déterminant est la taille de la collection. Ainsi une collection de taille réduite peut être indexée par un expert afin d'améliorer la représentativité des documents. Cependant, il est difficile d'indexer manuellement une collection de grande taille. Par conséquent, la solution la plus courante est une indexation automatique binaire ou pondérée des termes du texte.

L'indexation automatique peut se faire par une analyse statistique des distribution des termes dans les documents [][] ou par une analyse l'linguistique [][].

2.3.1 Choix des termes d'indexation

2.3.1.1 les Anti-dictionnaires

Un anti-dictionnaire correspond à une liste de termes qui ne sont porteur d'aucune information sémantique (appelé aussi mots vides). C'est le cas des propositions, pronoms, articles, etc. Ces mots sont généralement éliminés car ils se retrouvent dans tous les textes et sont indépendants des thèmes traités.

2.3.1.2 Radicalisation

Elle consiste à ne prendre en considération que les racines des termes rencontrés dans le texte. Ainsi le nombre total de termes du corpus se trouve réduit, car tous les mots ayant une même racine sont fusionnés en un même mot clés. La radicalisation à aussi l'avantage de permettre de retrouver des documents ne contenant pas les mêmes termes que la requête [Mot00](si les termes de la requête sont aussi radicalisés)

ça peut être une lemmatisation ou une stemmatisation. La lemmatisation consiste à bla bla bla ....., par exemple bla bla bla ...., . la stemmatisation consiste à bla bla bla ...., par exemple bla bla bla, l'algorithme le plus couramment utilisé est celui de Porter[].

2.3.1.3 La lois de Zipf

La loi de Zipf [Zip49] est une observation empirique sur la fréquence des mots dans un texte. Zipf a observé que la fréquence des termes suivent une certaine loi appelée principe du moindre effort (*Principle of Least Effort*). Il a ainsi montré qu'en classant les mots d'un texte par fréquence décroissante, on observe alors que la fréquence d'utilisation d'un mot est inversement proportionnelle à son rang[1] (c.f. figure 2.2). Plus formellement, si f_i est la fréquence du terme t_i et r_i son rang alors :

$$f_i * r_i \simeq C$$

où C est une constante.

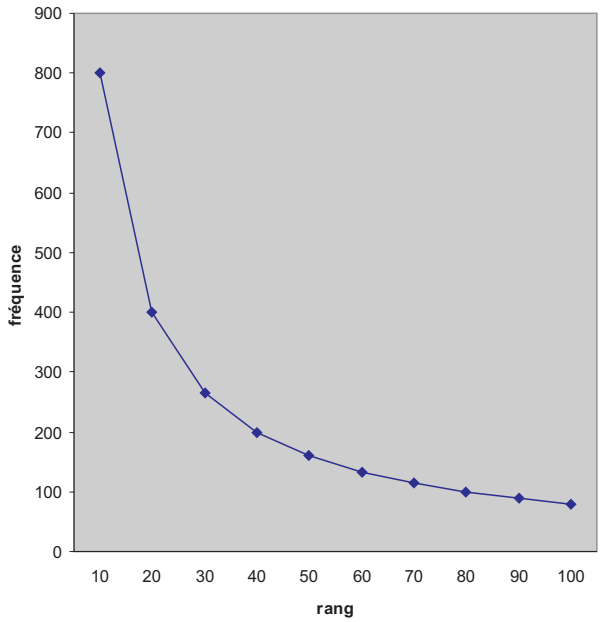

FIG. 2.2 – Représentation des termes par ordre décroissant de leurs fréquences

[1]Le terme les plus fréquent est de rang 1, alors que le moins fréquent est de rang maximal.

Zipf explique cette distribution des termes par le fait qu'il est plus facile pour un auteur de répéter certains termes que d'en utiliser des nouveaux.

Cette observation empirique permet de sélectionner les termes représentatifs des documents suivant leurs fréquences [Sal89] (absolue ou relative). En effet, la loi de *Zipf* montre que très peu de termes sont très fréquents mais représentent 50% du texte [Mot00]. Elle montre aussi que beaucoup de termes sont très peu fréquents. Ainsi, les termes très fréquents sont considérés comme non discriminants car ce types de termes se retrouvent dans tous les textes et sont indépendants des thèmes traités et les termes peu fréquents sont rarement utilisés et ne contribuent pas significativement au sens des textes [Luh58].

Notons que cette loi s'applique aussi bien aux fréquences relatives des termes (nombre d'occurrence dans un même document) qu'à leur fréquence absolue (ou fréquence en documents, i.e. nombre de documents dans lesquels ils apparaissent) [Poi99].

2.3.2 La pondération des termes

La pondération des termes est une fonction phare en RI, car elle a pour but d'attribuer des poids aux termes en fonction de leur importances. Ainsi il sera attribué plus de poids aux termes jugés plus importants que d'autres.

Le poids d'un terme dans un document peut dépendre du document (on parle alors de pondération locale),de la collection (on parle alors de pondération globale) ou de la normalisation.

1. pondération locale : elle mesure la représentativité du terme dans le document. Plusieurs fonction de pondération ont été proposées, on peut citer :
   - fonction binaire : le poids d'un terme t_i vaut 1 si il appartient au document et 0 sinon,
   - fonction fréquence : le poids d'un terme t_i correspond à sa fréquence d'occurrence dans le document tf_i,
   - fonction logarithme : le poids d'un terme t_i correspond au logarithme de sa fréquence dans le document $log(tf_i)$,
   - etc.
2. pondération globale : elle mesure la représentativité des termes dans l'ensemble des documents de la collection. En effet, un terme apparaissant dans beaucoup de documents est moins discriminant qu'un terme apparaissant dans peu de

documents. Le facteur de pondération global le plus utilisé est IDF (*Inverted Document Frequency*) où le poids d'un terme est inversement proportionnel à sa fréquence absolue DF (*Document Frequency*), plusieurs formules donnant IDF ont été proposées :

$$\begin{cases} IDF_i & = \frac{N}{df_i} \\ IDF_i & = log(\frac{N}{df_i}) \\ IDF_i & = log(\frac{N-df_i}{N}) \end{cases}$$

où N est le nombre total de documents dans la collection et df_i le nombre de documents où le terme t_i apparaît.

Souvent les pondérations locales et globales sont combinées, ainsi dans [RJ76] Robertson définit la pondération par TF-IDF qui est une combinaison de la pondération locale par fréquence TF (*Term Frequency*) et de la pondération global *IDF*. Ainsi le poids w_{ij} d'un terme t_i dans le document d_j pondéré par *TF-IDF* est calculé comme suit :

$$w_{ij} = TF_{ij} * IDF_i$$

3. Normalisation : consiste à tenir compte de la taille du document dans le processus de pondération des termes. L'objectif de la normalisation est de réduire l'avantage des documents longs sur les documents courts [SSMB95]. Différentes formules de normalisation ont été proposées [Sin97, RWHG94, RJ97, CCH92]. Par exemple la normalisation des poids des termes dans le modèle vectoriel peut se faire par une normalisation Euclidienne. Ainsi, soit w_{ij} le poids du terme t_i dans un document d_j, le nouveau poids normalisé w'_{ij} sera calculé comme suit :

$$w'_{ij} = \frac{w_{ij}}{\sqrt{\sum_{k=1}^{n} (w_{kj})^2}}$$

où n est le nombre de termes composant le vecteur document.

Une autre fonction de normalisation proposée par Roberstson [RWHG94] est formulée comme suit :

$$w_{ij} = nTF_i * IDF_i$$

avec

$$nTF_i = \frac{tf_{ij}}{tf_{ij} + 0.5 + 1,5 * \frac{lg_i}{lg_{moy}}}$$

$$idf_i = \frac{log(\frac{N+0.5}{df_i})}{log(N+1)}$$

où lg_i représente la longueur du document j en nombre de mots non vides et $lg_{moy} = \frac{\sum_i lg_i}{N}$

2.4 Les différents modèles de recherche d'information

2.4.1 Introduction

Les travaux de recherche effectués dans le domaine de la recherche d'information ont conduit à la proposition de nombreux modèles [Bou00] dont les principaux sont : les modèles booléens, les modèles vectoriels, le modèles probabilistes et les modèles connexionnistes.

Les modèles booléens se basent sur la théorie des ensembles. Dans ces modèles, la requête de recherche est exprimée par une liste des termes liés par des opérateurs booléens, dont principalement : la conjonction (ET), la disjonction (OU) et la Négation (NON) [LG00]. Ces modèles sont souvent utilisés par les moteurs de recherche sur Internet, car ils sont rapide et peuvent donc être utilisés en ligne. Plusieurs modèles booléens ont été proposés, à savoir le modèle booléen de base [], le modèle booléen étendu [SFW83], le modèle booléen flou [Sal89].

Dans les modèles vectoriels les documents et les requêtes sont représentés par des vecteurs de termes dans le même espace vectoriel. La pertinence d'une requête vis à vis d'un document est définie par des mesures de similarité vectorielles entre le vecteur représentant la requête et le vecteur représentant le document. Différents modèles vectoriels ont été proposés, tel que le modèle vectoriel binaire [Luh58], le modèle vectoriel standard [Sal71], le modèle LSI [DDL+90], etc.

Les modèles probabilistes quand à eux se basent sur la théorie des probabilités dans le processus de recherche d'information. Le principe de base de ces modèles est la présentation des résultats de recherche d'information dans un ordre basé sur la probabilité de pertinence d'un documents vis-à-vis d'une requête [Rob77]. Différents

modèles probabilistes ont été proposés : le modèle Robertson et Spark-Jones[RJ76], le modèle okapi [RWHG94], le modèle langage [PC98], le modèle inférentiel baysesien [TC91].

Dans les modèles connexionnistes les documents sont représentés par des neurones. Le processus de recherche d'information est basé sur la propagation des pondérations dans le réseau. Les résultats sont présentés à l'utilisateur en fonction du niveau d'activation des neurones documents. Il existe principalement deux catégories de systèmes connexionnistes : les systèmes à apprentissage supervisé [Bel89][Kwo89] et les les systèmes à apprentissage non supervisé [LF92][Poi99].

Dans la suite de cette section, nous décrivons les différentes classes de modèles de recherche d'information (booléens, vectoriels, probabilistes), ainsi que les principaux modèles dérivés.

2.4.2 Les Modèles Booléens

2.4.2.1 Le modèle booléen de base

Le modèle booléen est le plus simple des modèles de recherche. Il se base sur la théorie des ensembles et l'algèbre de BOOLE. Une requête de recherche est exprimée par des termes liées par les opérateurs ET, OU et NON [LG00]. Ce modèle utilise un appariement exact, i.e. un document est totalement pertinent ou pas du tout. Pour une requête de recherche q, le processus de recherche consiste à trouver les documents qui satisfassent aux conditions de la requête, i.e. les documents qui sont vrais pour la requête q.

Par exemple, pour la requête de recherche $q = (t1) \wedge (t2) \wedge (\neg t3)$, les documents considérés pertinents sont ceux contenant les termes t1 et t2 et ne contenant pas le terme t3.
La tableau 3.1 montre les valeurs de pertinence ($vp(q, d)$) associées à un document d pour chaque type de requête q.

Ce modèle est très simple à mettre en oeuvre, cependant il présente les inconvénients, suivants :

- il utilise un appariement exact, i.e. un document est totalement pertinent ou pas du tout, par conséquent il ne permet pas le classement des documents par pertinence décroissante,
- les termes de la requête sont pondérés de la même façon (0 ou 1), ainsi il n'existe pas de distinction entre termes discriminants et termes non discrimi-

q	$vp(q, d)$
t_i	=1 si $t_i \in d$ 0 sinon
$t_i \wedge t_j$	=1 si $t_i \in d$ et $t_j \in d$ 0 sinon
$t_i \vee t_j$	=1 si $t_i \in d$ et $t_j \in d$ 0 sinon
$\neg t_i$	=1 si t_i n'appartient pas à d 0 sinon

TAB. 2.1 – Évaluation des requêtes dans le modèle booléen de base

nants.

Afin d'y palier aux insuffisances du modèle booléen de base, des extensions de ce modèle ont été proposées : le modèle booléen étendu et le modèle booléen flou.

2.4.2.2 Le modèle booléen étendu

Afin de remédier aux insuffisances du modèle booléen de base, un modèle booléen étendu a été proposé par Salton [SFW83]. Dans celui-ci les termes dans les documents et les requêtes sont pondérés afin de permettre un appariement document requête approché.

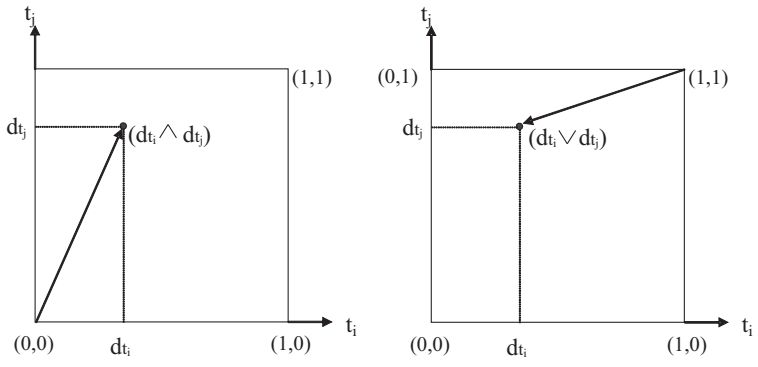

FIG. 2.3 – Principe du modèle booléen étendu

La figure 2.3 illustre le principe de fonctionnement du modèle booléen étendu.

Considérons les requêtes portant sur deux termes $q_1 = t_i \vee t_j$ et $q_2 = t_i \wedge t_j$ et un document $d = (d_{t_i}, d_{t_j})$, où d_{t_i}, d_{t_j} sont respectivement les poids des termes t_i et t_j dans d.

La similarité $sim(d, q_1)$ entre la requête $q_1 = t_i \vee t_j$ et le document d sera la distance normalisé entre le point (0,0) et le point $d = (d_{t_i}, d_{t_i})$ (formule 2.1). De manière analogue la similarité $sim(d, q_2)$ entre la requête $q_2 = t_i \wedge t_j$ et le document d sera la distance inverse entre le point (1,1) et et le point $d = (d_{t_i}, d_{t_i})$ (formule 2.2).

$$sim(D, d_{t_i} \vee d_{t_j}) = \sqrt{\frac{(d_{t_i} - 0)^2 + (d_{t_j} - 0)^2}{2}} \quad (2.1)$$

$$sim(D, d_{t_i} \wedge d_{t_j}) = 1 - \sqrt{\frac{(1 - d_{t_i})^2 + (1 - d_{1-t_j})^2}{2}} \quad (2.2)$$

Dans le cas de requêtes pondérés les formules 2.1 et 2.2 peuvent être étendues de la manière suivante :

$$sim(D, d_{t_i} \vee d_{t_j}) = \sqrt{\frac{q_{t_i}^2 d_{t_i}^2 + q_{t_j}^2 d_{t_j}^2}{q_{t_i}^2 + q_{t_j}^2}}$$

$$sim(D, d_{t_i} \wedge d_{t_j}) = 1 - \sqrt{\frac{q_{t_i}^2 (1 - d_{t_i})^2 + q_{t_j}^2 (1 - d_{1-t_j})^2}{q_{t_i}^2 + q_{t_j}^2}}$$

2.4.2.3 Le modèle booléen flou

Contrairement à la logique binaire, la logique flou [Zad65] est une logique fondée sur des variables pouvant prendre d'autre valeurs outre les valeurs "vrai" ou "faux" (0 ou 1), des valeurs intermédiaires comprise dans l'intervalle [0, 1].

Le modèle booléen flou est basé sur la théorie des ensemble flous. Il a été introduit par Salton en 1989 [Sal89]. Dans ce modèle et contrairement au modèle booléen de base, les résultats des opérations logiques, i.e. le coefficient de similarité entre la requête est un document, peuvent prendre des valeurs comprises dans l'intervalle [0, 1].

Plusieurs modèles booléens flous ont été proposés [OMK91][Sal89]. Ainsi, dans [Sal89], chaque document d_i est considéré comme étant un ensemble flou et chaque terme t_i comme objet de d_i. On peut alors définir une fonction d'appartenance $f_{d_i}(t_j)$ du terme t_j dans l'ensemble d_i. Ainsi un document d_i peut être représenté par un ensemble de termes pondérés :

$$d_i = \{(t_1, w_{t_1})..(t_n, w_{t_n})\}$$

où $f_{d_i}(t) = w_{t_i}$ est un entier compris entre 0 et 1, il représente le degré d'appartenance du terme t_i dans l'ensemble flou d_i.

Une requête q est une proposition flou dont on peut calculer son degré d'appartenance $f_{d_i}(q)$ à chaque ensemble flou d_i (c.f. tableau 2.2).

q	$f_{d_i}(q))$
t_i	$f_{d_i}(t_i))$
$t_i \wedge t_j$	$min(f_{d_i}(t_i)), f_{d_i}(t_j)))$
$t_i \vee t_j$	$max(f_{d_i}(t_i)), f_{d_i}(t_j)))$
$\neg t_i$	$1 - f_{d_i}(t_i))$

TAB. 2.2 – Exemple d'évaluation de requêtes dans le modèle booléen flou

2.4.3 Les Modèles vectoriels

2.4.3.1 Le Modèle vectoriel standard

L'idée de représenter les documents par des vecteurs de termes a été introduite par Luhn durant les année 1950 [Luh58]. Ensuite elle a été reprise est développée par Salton et son équipe dans leur projet SMART [Sal71].

Dans le modèle vectoriel standard, chaque document d est représenté par un vecteur à n dimensions $(w_1, .., w_n)$, où w_i est le poids du terme t_i dans le document d. Cette représentation requiert la définition de l'ensemble des termes d'indexation et une méthode de pondération des termes comme nous l'avons présenté dans la section 2.3.

La pertinence d'un document par rapport à une requête dépend de la position de leur vecteur respectifs dans l'espace vectoriel (une requête est aussi un document et peut donc être converti en un vecteur). Ainsi, plus un vecteur document est similaire à un vecteur requête plus ce document est pertinent.

Il existe différentes formules pour calculer la similarité d'un document vis à vis d'une requête. Soit $\vec{d_i} = (w_{d_{i1}}, w_{d_{i2}}, .., w_{d_{in}})$ la représentation vectorielle du document d_i et $\vec{q_i} = (w_{q_{i1}}, w_{q_{i2}}, .., w_{q_{in}})$ la représentation vectorielle de la requête q_i. Les mesures de similarité les plus répandues sont :

– Le produit scalaire :

$$sim(\vec{d_i}, \vec{q_i}) = \sum_{j=1}^{n} (w_{d_{ij}}.w_{q_{ij}})$$

– la formule du cosinus : c'est la mesure de similarité la plus utilisée, elle consiste à calculer le cosinus de l'angle entre le vecteur document et le vecteur requête :

$$sim(\vec{d_i}, \vec{q_i}) = \frac{\sum_{j=1}^{n} (w_{d_{ij}}.w_{q_{ij}})}{\sqrt{\sum_{j=1}^{n} (w_{d_{ij}}^2.w_{q_{ij}}^2)}}$$

– La mesure de Jacard :

$$sim(\vec{d_i}, \vec{q_i}) = \frac{\sum_{j=1}^{n} (w_{d_{ij}}.w_{q_{ij}})}{\sum_{j=1}^{n} w_{d_{ij}}^2 + \sum_{j=1}^{n} w_{q_{ij}}^2 - \sum_{j=1}^{n} (w_{d_{ij}}.w_{q_{ij}})}$$

– La mesure de Dice :

$$sim(\vec{d_i}, \vec{q_i}) = 2\frac{\sum_{j=1}^{n} w_{d_{ij}}.w_{q_{ij}}}{\sum_{j=1}^{n} (w_{d_{ij}}^2 + w_{q_{ij}}^2)}$$

Le principale inconvénient du modèle vectoriel standard est l'indépendance des termes. Des modèles vectoriels palliant cet inconvénient ont été proposés, tel que le modèle vectoriel généralisé [WZW85] et le modèle LSI (*Latent Semantic Indexing*) [DDL+90].

2.4.3.2 Le modèle conceptuel

Le modèle vectoriel décrit précédemment peut être appliqué de manière identique à un espace constitué non pas de termes mais de concepts [Rog52, Cha90]. Ainsi, les documents et les requêtes seront représentés par des vecteurs dans cet espace conceptuel.

Plus formellement, soit $C = (c_1, c_2, ..c_n)$ un ensemble fini de n concepts. Un vecteur conceptuel V est une combinaison linéaire des éléments de C. Chaque mot de la langue a une représentation vectoriel dans l'espace conceptuel, par conséquent la représentation d'un document dans l'espace conceptuel consiste en l'addition des vecteurs conceptuels des termes constituant le document.

Plus généralement, toute méthode définie avec des termes peut être mises en oeuvre avec des concepts à condition que l'on dispose d'un moyen pour faire correspondre les termes et les concepts [Lef00].

2.4.3.3 Le modèle LSI

Le modèle *LSI* (*Latent Semantic Indexing*) [DDL+90] est une variante du modèle vectoriel standard, où les documents sont représentés dans un espace de dimension réduite issu de l'espace initial des termes d'indexation.

LSI utilise une décomposition en valeurs singulières sur la matrice représentative du corpus (documents × termes), cette décomposition permet d'extraire les principales associations entre les termes d'un document.

Plus formellement, la représentation des documents et requêtes dans *LSI* s'effectue de la manière suivante :

- iInitialement les documents sont représentés par des vecteurs de termes de dimension t. L'ensemble des vecteurs de documents sont représentés par une matrice X de dimension $t \times d$ où d est le nombre de documents de la collection et t le nombre de termes d'indexation.
- Ensuite, une décomposition en valeur singulière SVD (*Singular Value Decomposition*) est effectuée pour créer un nouvel espace vectoriel :

  $$X = T_0.S_0.D_0^t$$

  avec :
  T_0 : matrice orthogonale de dimension $t \times m$
  D : matrice orthogonale de dimension $m \times d$
  S_0 : matrice diagonale de dimension $m \times m$, où les valeurs de la diagonale sont les valeurs propres de X
- LSI considère que les petites valeurs propres de la matrice S_0 sont porteuses de bruit et leur élimination minimise la perte d'information. Par conséquent seuls les k premiers vecteurs propres de la matrice S_0 sont pris en compte. Ainsi, les matrices S_0, T_0 et D_0^t nettoyées deviennent repectivement S, T et D^t et la matrice X devient $X^{'}$ tel que :

  $$X^{'} = T.S.D^t \simeq T_0.S_0.D_0^t$$

- Enfin, une requête X_q est représentée par le vecteur q dans le nouveau espace comme suit :

  $$q = X_q.T.S^{-1}$$

Ce modèle représente les documents et requêtes dans un espace réduit de dimension k, cependant le nouvel espace de représentation est difficilement interprétable car cet espace n'est pas un ensemble de termes.

Dans le cadre de la recherche d'information les performances de ce modèle sont très supérieurs à celle du modèle vectoriel classique pour des corpus de petit taille, cependant quand la taille de corpus augmente la différence semble diminuer.

Des variantes [Hof01] et des applications de ce modèle ont été proposés dans denombreux domaines d'analyse de documents textuels : recherche d'information [FDD+88][DLL96], filtrage [FD92] [Hof04], classification [LZO03][ZH01], etc.

2.4.4 Les Modèles connexionnistes

L'utilisation des réseaux de neurones dans le domaine de la recherche d'information est apparue dans les années quatre-vingt avec les travaux de M. Mozer [Moz99].

Un système de recherche d'information connexionniste est constitué de neurones formels, généralement organisés en couches et de liens pondérés reliant les neurones entre les différentes couches. Les neurones formels sont construits à partir des représentations initiales des documents.

Le processus de recherche d'information est basé sur la propagation des pondérations dans le réseau, i.e. depuis les neurones descriptifs de la requête vers ceux des documents. Les résultats sont présentés à l'utilisateur en fonction du niveau d'activation des neurones documents. Les liens du réseau étant modifiables par apprentissage ce qui les rendent particulièrement adaptés à la ré-injection de la pertinence utilisateur.

Les systèmes de recherche d'information connexionniste permettent de combler les insuffisances de certains modèles vectoriels. En effet les réseaux de neurones permettent de représenter différentes associations :

- relations entre termes (synonymie, voisinage, ..),
- relations entre documents (référence, similitude, ..),
- relation entre termes et documents (fréquence,..etc),

De nombreux modèles basés sur le principe des réseaux de neurones ont été proposés dans le contexte de la recherche d'information [Bou92] [Mot94][WCY93][Cre97] [Bel89].

Il existe différentes représentations pour modéliser un système de recherche d'information connexionniste. Cependant les systèmes de recherche d'information utilisant les réseaux de neurones peuvent être repartis en deux catégories principales :

- **Les systèmes à apprentissage supervisé**

Généralement, construit autour d'un réseau en couches, ils sont constitués au minimum de deux couches, une recevant les entrées (requêtes de recherche) et une couche de sortie fournissant les résultats (activation des documents). En plus des deux couches précédentes il peut exister des couches intermédiaires ou cachés.
Par exemple, les systèmes proposés dans [Bel89][WH91], sont constitués de deux couches : une couche "terme d'indexation" et une couche "documents". D'autres systèmes sont quand à eux constitués de plusieurs couches, ainsi dans le système proposé par Kwok [Kwo89] il existe trois couches, une dédiée aux requêtes, une aux termes d'indexation et une autre aux documents .
Le processus de recherche d'information est basé sur la propagation des pondérations dans le réseau depuis les neurones descriptif de la requête vers ceux des documents.

– **Les systèmes à apprentissage non supervisé**
Contrairement aux systèmes à apprentissage supervisé, seules les entrées modifient les poids des connexions des neurones. Généralement basés sur les cartes auto-organisatrice de Kohonen (*Self Organizing Map*) [Koh89], dans ces systèmes les réseaux sont utilisés pour la classification des documents [LF92][Lin97][KKLH96][CSO96][Poi99].
Ces systèmes sont constitués de deux couches. La première, appelée couche d'entrée, comporte autant de neurones que les données ont de composantes. La deuxième couche, appelée couche de sortie, chaque neurone de celle-ci est connecté à tous les neurones de la couche d'entrée.
L'apprentissage consiste à calculer à partir d'un vecteur d'entrée, une distance avec tous les neurones de la couche. Ensuite le neurone de la couche de sortie dont la distance est la plus proche du vecteur d'entrée sera sélectionné et les poids de ce neurone sont actualisés de façon à être plus proche des neurones d'entrées.

2.4.5 Les Modèles probabilistes

Les modèles probabilistes se basent sur la théorie des probabilités dans le processus de recherche d'information. Le premier modèle a été proposé par Maron et Kuhn [MK60] . Le principe de base de ces modèles est la présentation des résultats de recherche d'information dans un ordre basé sur la probabilité de pertinence d'un documents vis-à-vis d'une requête [Rob77].

Différents modelés probabilistes ont été proposés [RJ76, RRP81][RW94][PC98][Hie02, HRZ04], nous présentons dans les sous-sections suivantes quelques modèles les plus représentatifs, à savoir : le modèle Robertson et Spark-Jones[RJ76] et le modèle okapi [RWHG94].

2.4.5.1 Modèle de Robertson et Spark-Jones

Le modèle de Robertson et Spark Jones [RJ76], repose sur le principe d'ordre des probabilités (*Probability Ranking Principle*) [Rob77]. La similarité entre un document et une requête est calculée à partir de l'estimation de la probabilité que le document a d'être pertinent et non pertinent pour la requête.

L'idée de base du modèle est de tenter de déterminer les probabilités $P(R|d)$ (probabilité qu'un document d soit pertinent (noté R pour *relevant*) et $P(\bar{R}|d)$ probabilité qu'un document d soit non pertinent (noté $\bar{R}$ pour *non-relevant*) pour une requête donnée.

Les documents pourront ensuite être classés selon ces deux probabilités :

$$sim(d_j, q) = \frac{P(R|d)}{P(\bar{R}|d)} \tag{2.3}$$

En appliquant le théorème de bayes, les probabilités $P(R|d)$ et $P(\bar{R}|d)$ peuvent être réécrites par :

$$P(R|d) = \frac{P(D|R)P(R)}{P(d)} \tag{2.4}$$

$$P(\bar{R}|d) = \frac{P(D|\bar{R})P(\bar{R})}{P(d)} \tag{2.5}$$

Ainsi, en utilisant les équations 2.4 et 2.5 l'équation 2.3 devient :

$$sim(d_j, q) = \frac{P(D|R)P(R)}{P(D|\bar{R})P(\bar{R})} \tag{2.6}$$

Comme pour la même requête $P(R)$ et $P(\bar{R})$ sont des constantes, $sim(d_j, q)$ peut être ré-exprimée comme suit :

$$sim(d_j, q) = \frac{P(D|R)}{P(D|\bar{R})} \tag{2.7}$$

Un document d_j peut être décomposeé en un ensemble d'événements dénotant la présence ou l'absence de termes dans le document. Ainsi $sim(d_j, q)$ peut être ré-exprimée par :

$$sim(d_j, q) = \frac{P(t_1 = x_1, t_2 = x_2, .., t_n = x_n|R)}{P(t_1 = x_1, t_2 = x_2, .., t_n = x_n|\bar{R})} \tag{2.8}$$

où $x_i = 0$ ou 1 et $t_i = x_i$ correspond à la présence ($x_i = 1$) ou à l'absence ($x_i = 0$) du terme t_i dans le document d_j.

En faisant l'hypothèse que les occurrences des différents termes sont indépendantes l'équation 2.8 peut être remplacée par :

$$sim(d_j, q) = \prod_{t_i \in d_j} \frac{P(t_i = x_i|R)}{P(t_i = x_i|\bar{R})} = \prod_{t_i \in d_j} \frac{P(t_i|R)}{P(t_i|\bar{R})} \times \prod_{t_i \notin |d_j} \frac{P(\bar{t_i}|R)}{P(\bar{t_i}\bar{R})} \tag{2.9}$$

$P(t_i|R)$ (respectivement $P(t_i|\bar{R})$) représente la probabilité que le terme t_i soit présent dans un document pertinent (respectivement non pertinent) et $P(\bar{t_i}|R)$ (respectivement $P(\bar{t_i}|\bar{R})$) représente la probabilité que le terme t_i ne soit pas présent dans un document pertinent (respectivement non pertinent).

En effectuant quelques transformations (logarithme, élimination des constantes, etc), l'équation 2.9 devient :

$$sim(d_j, q) = \sum_{t_i \in d_j} log \frac{P(t_i|R)(1 - P(t_i|\bar{R}))}{P(t_i|\bar{R})(1 - P(t_i|R))} + \sum_{t_i \notin d_j} log \frac{P(t_i|R)(1 - P(t_i|\bar{R}))}{P(t_i|\bar{R})(1 - P(t_i|R))} \tag{2.10}$$

Nous avons aussi :

$$w_i = log \frac{P(t_i|R)(1 - P(t_i|\bar{R}))}{P(t_i|\bar{R})(1 - P(t_i|R))} \tag{2.11}$$

w_i peut être considéré comme le poids du terme t_i dans le document d_j associé à la requête q.

L'estimation des probabilités $P(t_i|R)$ et $P(t_i|\bar{R})$ s'effectue sur un ensemble d'échantillons de documents déjà jugés pour une requête. Ainsi, soit pour une requête donnée q et un terme t_i :

- N le nombre de documents de l'échantillon ;
- R le nombre de documents (de l'échantillon) pertinents par rapport à q
- n_i le nombre de documents (de l'échantillon) contenant t_i
- r_i le nombre de documents (de l'échantillon) pertinents contenant t_i

les probabilités $P(t_i|R)$ et $P(t_i|\bar{R}$ sont alors :

$P(t_i|R) = r_i/R$

$P(t_i|\bar{R} = n - r_i/N - R$

Le poids w_i du terme t_i peut alors s'écrire :

$$w_i = log \frac{\frac{r_i}{R-r}}{\frac{n_i-r}{N-n_i-R+r}} \tag{2.12}$$

Pour éviter les problèmes de valeurs nulles ($R - r_i = 0$ ou $N - n_i - R + r = 0$) la valeur de 0.5 peut être ajouté. Ainsi l'équation 2.12 devient :

$$w_i = log \frac{\frac{r_i}{R-r+0.5}}{\frac{n_i-r}{N-n_i-R+r+0.5}} \tag{2.13}$$

L'hypothèse de l'indépendance des occurrence des termes utilisés afin de simplifier le calcul n'est pas réaliste. Cependant si on veut tenir compte de toutes les dépendances alors le calcul des probabilités $P(D|R)$ et $P(D|\bar{R})$ sera très complexe. Ainsi des approches intermédiaires tenant compte partiellement des dépendances des termes ont été proposées [Coo91, CCG94][Rij77].

2.4.5.2 Modèle Okapi

Le modèle proposé par Robertson et Sparck Jones ne considère que la présence ou l'absence d'un terme dans un document. Il ne tient pas compte de la fréquence d'occurrences des termes dans le document.

S'inspirant des travaux de Harter [Har75], le modèle probabiliste Okapi proposé par Roberston et al [RRP81, RWHG94] tient compte de la fréquence d'occurrences des termes dans le document pour l'estimation des probabilités $P(t_i|R)$ et $P(t_i|\bar{R})$. Ainsi dans Okapi l'équation 2.11 devient :

$$w_i = log \frac{P(tf(t_i)|R)(1 - P(tf(t_i)|\bar{R}))}{P(tf(t_i)|\bar{R})(1 - P(tf(t_i)|R))} \tag{2.14}$$

où $tf(t_i)$ représente la fréquence du terme t_i dans le document d.

Pour prendre en compte les fréquences des termes, il est nécessaire d'avoir un modèle de la distribution des fréquence dans les documents. Dans Okapi, les fréquences des termes ont été modélisées pas une loi de poisson.

Les distributions des termes sont construites en se basant sur l'hypothèse qu'un terme est relié à un seul thème et qu'un document parle ou ne parle pas de ce thème. Cependant un document ne parlant pas de ce thème peut utiliser ce terme.

Ainsi deux distributions de termes suivant la loi de poisson sont construites :

- une distribution pour les documents parlant du thème
- une distribution pour les autres documents

La propriété de parler d'un thème associée à un terme est nommé élitisme (*elitness*). Ainsi dans Okapi une hypothèse d'indépendance est faite sur l'élitisme des termes, on a alors :

$$\begin{cases} P(tf(t_i)|\bar{R}) = p(tf(t_i)|E)p(E|\bar{R}) + p(tf(t_i)|\bar{E})p(\bar{E}|\bar{R}) \\ P(tf(t_i)|R) = p(tf(t_i)|E)p(E|R) + p(tf(t_i)|\bar{E})p(\bar{E}|R) \end{cases}$$

où E représente l'ensemble des documents élites du terme t_i

La pondération w_i du terme t_i est alors :

$$w_i = \frac{(p'\lambda^{tf_i}e^{-\lambda} + (1-p')\mu^{tf_i}e^{-\mu})(q'e^{-\lambda} + (1-q')e^{-\mu})}{(p'\lambda^{tf_i}e^{-\lambda} + (1-p')\mu^{tf_i}e^{-\mu})(q'e^{-\lambda} + (1-q')e^{-\mu})} \tag{2.15}$$

où :

λ représente l'espérance de la loi de poisson pour l'ensemble des documents élites,

μ représente l'espérance de la loi de poisson pour l'ensemble des documents non-élites,

$p' = p(E|R)$: est la probabilité q'un document pertinent soit élite pour t_i

$q' = p(E|\bar{R})$: est la probabilité q'un document non pertinent soit élite pour t_i

Dans [Okapi 3] une approximation de l'équation 2.15 est faite par une fonction simple ayant le même comportement général (nulle si $tf = 0$, monotone croissante avec tf_i et possédant un maximum asymptotique). La pondération w_i du terme t_i sera alors :

$$w_i = \frac{tf_i}{k_1 + tf_i} \times log\frac{(r_i + 0.5)(R - r + 0.5)}{(n_i - r + 0.5)(N - n_i - R + r + 0.5)}$$

où k_1 une constante permettant d'accroître ou de diminuer l'influence de la fréquence du terme tf_i sur w_i.

2.5 L'expansion de requêtes

L'un des problèmes majeurs de la recherche d'information est la formulation des requêtes. Blair et Maron [BM85] ont montré que la faible performance des systèmes de recherche d'information est dû généralement à l'incapacité des utilisateurs de formuler les requêtes adéquates. En effet, la requête initiale de l'utilisateur est souvent exprimées par une liste de termes souvent très réduite qui exprime mal les besoins en information de l'utilisateur. Pour remédier à ce problème, une solution

consiste à étendre automatiquement la requête initiale afin d'améliorer la qualité des documents retrouvés.

La reformulation d'une requête consiste en l'ajout et/ou retrait de termes de la requête initiale [Eft00]. Il existe deux classes d'approches pour la reformulation de requête [Teb04] :

1. La re-injection de la pertinence utilisateur : elle consiste à modifier la requête utilisateur à l'aide des documents jugés pertinents et/ou non pertinents par l'utilisateur.
2. L'utilisation structures de connaissance : elle consiste à reformuler la requête au travers de liens sémantiques ou statistiques établis entre termes. Ces liens sont construit manuellement par un expert ou de manière automatique. Dans ce dernier cas, on parle alors de reformulation de requêtes par :
   (a) Analyse locale : l'expansion de requêtes se fait automatiquement par l'analyse des documents retrouvés.
   (b) Analyse globale : l'expansion de requêtes se fait automatiquement par l'analyse de l'ensemble des documents de la collection.

Dans les sous sections suivantes, nous commençons par présenter les méthodes principales d'expansion de requête par ré-injection de pertinence utilisateur. Ensuite nous nous intéressons aux méthodes d'expansion basées sur l'analyse locale et globale.

2.5.1 La ré-injection de la pertinence utilisateur

Elle consiste à modifier la requête utilisateur et cela à travers des documents jugés pertinents et/ou non pertinents par l'utilisateur.

Basé sur les jugés pertinents et/ou non pertinents par l'utilisateur, cette reformulation peut se faire par une re-pondération des termes de la requête et/ou l'ajout (et/ou retrait) de termes contenus dans les documents pertinents (non pertinents).

Cette opération s'effectue de la manière suivante :

1. l'utilisateur effectue une première requête de recherche,
2. le système retourne un ensemble de documents,
3. l'utilisateur indique parmi les documents retournés ceux qui sont pertinents et/ou non pertinents,

4. le système modifie alors automatiquement la requête de départ en fonction des jugements de l'utilisateur.

Ainsi, la requête est modifiée de manière "à ressembler" aux documents jugés pertinents et "à s'éloigner" des documents non pertinents. Les expérimentation de ré-injection de la pertinence utilisateur effectuées dans le cadre du système Smart [Sal71] et dans le modèle probabiliste [RJ76] ont montré une amélioration significative des résultats pour les collections documentaire de petite taille. Cependant, l'inconvénient majeur de cette méthodes est l'implication de l'utilisateur.

Plusieurs techniques de ré-injection de la pertinence utilisateur ont été proposées dans la littérature [Roc71][SB90][DD80][SVF84][RRP81][RJ76][Kwo89]. Dans les sous sections suivantes nous présentons quelques techniques de ré-injection de pertinence proposées pour le modèle vectoriel et le modèle probabiliste.

2.5.1.1 La ré-injection de la pertinence utilisateur dans le modèle Vectoriel

Dans le modèle vectoriel, la ré-injection de la pertinence utilisateur se fait généralement par l'ajout au vecteur requête initial des poids des termes des document jugés pertinents et la soustraction au vecteur requête initial des poids des termes des documents non pertinents.

L'algorithme de Rochhio développé au milieu des années soixante [Roc71][Roc66], est l'un des plus utilisés, la forme standard de l'algorithme de Rochhio est la suivante :

$$\vec{q'} = \alpha\vec{q} + \frac{\beta}{|D_r|}\sum_{\forall\vec{d_j}\in D_r}\vec{d_j} - \frac{\gamma}{|D_n|}\sum_{\forall d_j\in\vec{D}_n}\vec{d_j}$$

où :
$\vec{q'}$: le nouveau vecteur requête,
$\vec{q}$: le vecteur de la requête initiale,
D_r : l'ensemble des documents jugés pertinents par l'utilisateur,
D_n : l'ensemble des documents jugés non pertinents par l'utilisateur
$|D_r|$, $|D_n|$: représentent respectivement le nombre de documents des ensembles D_r et D_n,
α, β, γ : paramètres constantes,

D'autres méthodes de reformulation de requêtes se sont inspirés de l'algorithme de Rocchio, à savoir :

Méthode de régulation de Ide [Ide71] :

$$\vec{q'} = \alpha\vec{q} + \beta \sum_{\forall \vec{d_j} \in D_r} \vec{d_j} - \gamma \sum_{\forall d_j \in \vec{D_n}} \vec{d_j}$$

Méthode Ide_Dec_Hi :

$$\vec{q'} = \alpha\vec{q} + \beta \sum_{\forall \vec{d_j} \in D_r} \vec{d_j} - \gamma\, max_{non-pertinents}(\vec{d_j})$$

où $max_{non-pertinents}(\vec{d_j})$, désigne les vecteurs de documents non pertinents de score le plus élevé.

2.5.1.2 La ré-injection de la pertinence utilisateur dans le modèle probabiliste

Dans le modèle probabiliste de Robertson et Spark Jones [Rob77] la similarité du document d_j à une requête q donnée est formulée par :

$$sim(d_j, q) = \sum_{t_i \in d_j} log \frac{P(t_i|R)(1 - P(t_i|\bar{R}))}{P(t_i|\bar{R})(1 - P(t_i|R))} + \sum_{t_i \notin d_j} log \frac{P(t_i|R)(1 - P(t_i|\bar{R}))}{P(t_i|\bar{R})(1 - P(t_i|R))} \quad (2.16)$$

où $P(t_i|R)$ (resp. $P(t_i|\bar{R})$) représente la probabilité que le terme t_i soit présent dans un document pertinent (resp. non pertinent).

Le calcul de $sim(d_j, q)$ est conditionné par une connaissance de $P(t_i|R)$ et $P(t_i|\bar{R})$. Une solution possible consiste à estimer ces probabilités au travers des documents jugés par l'utilisateur (documents provenant d'une recherche initiale).

Cette opération peut être résumée comme suit :

1. affecter des valeurs initiales aux probabilités $P(t_i|R)$ et $P(t_i|\bar{R})$,
2. l'utilisateur effectue une première requête de recherche,
3. le système retourne un ensemble de documents,
4. l'utilisateur indique parmi les documents retournés ceux qui sont pertinents et/ou non pertinents,
5. le système modifie alors les valeurs des $P(t_i|R)$ et $P(t_i|\bar{R})$ en fonction des jugements de l'utilisateur,
6. le système retourne un ensemble de documents.

2.5.2 Analyse locale

Les techniques basées sur l'analyse locale permettent d'identifier les relations entre termes afin d'enrichir les requêtes par l'analyse des documents retrouvés les mieux classés [AF77, BSAS94, CH79, XC00, RWJ+95, CCCH02]. Dans [BSAS94], les auteurs proposent une technique qui suppose que les premiers documents retrouvés sont pertinents, ensuite la requête est enrichie suivant la méthode standard de ré-injection de la pertinence [Roc71]. Une méthode similaire est utilisée dans [CH79], où les premiers documents retrouvés sont utilisés pour ré-estimer les probabilités des termes.

Nous décrivons dans les sous sections suivantes trois méthodes d'expansion de requête par analyse locale : la méthode d'expansion par classification locale proposée par Attar et Frankel [AF77], la méthode d'expansion par ré-injection locale (*local feedback*)[BSAS94] et enfin la méthode d'expansion par analyse du contexte local [XC96][XC00].

2.5.2.1 Classification locale

La méthode d'expansion de requête par classification locale proposée par Attar et Frankel [AF77] est la première méthode à utiliser les résultats de la recherche pour enrichir la requête de manière automatique [XC00]. Elle consiste à étendre la requête initiale à partir d'une classe de termes extraite des résultats de la recherche de la requête initiale. Le processus d'expansion par classification locale peut se résumer comme suit :

1. trouver un ensemble de documents en effectuant une recherche avec la requête initiale,
2. classer les termes des documents retrouvés (i.e. plusieurs formules pour calculer les distances entre termes ont été proposés par les auteurs [AF77]),
3. étendre la requête initiale en utilisant les classes de documents, pour cela chaque terme t_i de la requête initiale, les m termes les plus similaires et de la même classe que t_i sont considérés.

Les résultats expérimentaux obtenus par Attar et Fraenkel ont montré une amélioration des résultats. Cependant les collections de documents utilisés sont très petites et ne permettent pas de tirer des conclusion définitives.

2.5.2.2 La ré-injection locale

L'expansion de la requête par la ré-injection locale (*local feedback*)[BSAS94][BSMS96] est semblable aux méthodes d'expansion par ré-injection de la pertinence utilisateur [Roc71], la différence majeur est que les k premiers documents retrouvés sont supposés pertinents. Les résultats obtenus sont très encourageants, cependant le problème majeur de cette méthode est que les résultats peuvent être sensiblement dégradés si parmi les k premiers documents retrouvés peu d'entre eux sont pertinents [Xu97].

2.5.2.3 Analyse du contexte local

Dans cette sous section nous allons décrire la technique d'expansion de requête par analyse local proposée par Xu et Croft appelé Analyse du Contexte Local (*Local Context Analysis*) [XC96][XC00]. Comme toute méthode d'analyse locale, la méthode d'analyse du contexte local utilise les n premiers documents retrouvés pour reformuler la requête. Cependant au lieu d'utiliser les documents en entier, seul un passage par document est utilisé. Un passage correspond à une fenêtre de mots d'un document.

Le but du choix d'un passage par document est d'éviter le problème potentiel d'utiliser dans le processus expansion de requêtes des termes provenant des parties indépendantes d'un longs document, car souvent dans les long documents il existe plusieurs parties parlant de thématiques différentes.

Le principe de la méthode est le suivant :

1. sélectionner les n premiers documents retrouvés par la requête initiale,
2. sélectionner un passage par document (un passage est une fenêtre de mots, sa longueur optimale fixée empiriquement est de 300 mots [XC00]),
3. extraire à partir de ces passages les concepts d'expansion (pour plus de détails voir *PhrasePhinder* [JC94]). Un concept peut être un nom ou un groupe nominal. Un concept c est représenté par un ensemble de tuples $\{< t_1, a_1 >, < t_1, a_1 >, ..\}$, où t_i est un terme en co-occurrence avec le concept c, et a_i est le nombre de co-occurences entre t_i et c. L'extraction des concepts est faite via l'utilisation de l'étiqueteur morpho-syntaxique *Jtag* [XBC94]
4. calculer la similarité $f(c, Q)$ via l'équation 2.17) entre la requête initiale Q et chacun des concepts extraits c,
5. ordonner les concepts selon leur similarité $f(c, Q)$ avec la requête initiale q,

6. sélectionner les k concepts les plus similaires à la requête pour l'expansion de la requête initiale (la valeur optimale de k fixée empiriquement est 30)

$$f(c, Q) = \prod_{w_i \in Q} (\delta + co\_degree(c, w_i))^{idf(w_i)} \tag{2.17}$$

où :

δ est une constante qui permet d'éviter les valeurs nulles,

$idf(w_i) = min(1.0, log_{10}(N/N_{w_i})/0.5$ où :

N représente le nombre de passages, i.e. nombre de documents retrouvés,

N_{w_i} représente le nombre de passages contenant le terme w_i,

$co\_degree(c, w_i)$:est la fonction de calcul de similarité, i.e. degré de co-occurrence, entre le concept c est le terme w_i avec :

$$co\_degree(c, w_i) = log_{10}(co(c, w_i) + 1)\frac{idf(c)}{log_{10}(n)}$$

où :

$idf(c) = min(1.0, log_{10}(N/N_c)/0.5)$ et N_c représente le nombre de passages contenant le concept c

$co(c, w_i) = \sum_{p \in Q} tf(c, p)tf(w_i, p)$ où :

$tf(c, p)$ et $tf(w_i, p)$ représentent respectivement la fréquence du concept c et du terme w_i dans le passage p.

Notons que une fois les concepts choisis, la requête est reformulé de manière brute, c'est à dire que les concepts sont rajoutés à la requête initiale tel que saisie par l'utilisateur.

2.5.3 Analyse globale

Les méthodes d'expansion de requête par analyse locale décrites précédemment permettent l'expansion au travers l'analyse des documents retrouvés. Une approche alternative appelé analyse globale consiste à étendre la requête au travers de l'analyse de l'ensemble des documents de la collection.

Différentes approches ont été proposées pour l'expansion de requêtes par analyse globale, ainsi l'analyse de la collection documentaire peut se faire par : classification des termes [JJ70][CY92], classification des documents [CY92], analyse des collocations [VRJ03, WV05], thésaurus de similarité [QF93, JC94], etc.

Nous décrivons dans les sous sections suivantes deux méthodes phares d'expansion de requêtes par analyse globale : la méthode d'expansion par thésaurus de similarité proposé par Qui et Frei [QF93] et la méthode d'expansion par classification de documents appelé thésaurus statistique globale proposé par Crouch et Yang [CY92].

2.5.3.1 Thésaurus de similarité

Les méthodes d'expansion de requêtes basés sur un thésaurus de similarité [QF93] consistent à rajouter aux termes de la requête des termes issus du theésaurus de similarité. Globalement ceci revient à calculer des valeurs de similarité entre les termes de la requête et les autres termes d'indexation en utilisant les valeurs de similarités entre termes données par le thésaurus. Ensuite les k termes les plus similaires à la requête sont rajoutés à la requête initiale.

Un thésaurus de similarité est une matrice de similarité terme-terme [SK92]. Pour calculer la similarité entre les différents termes d'indexation, chaque terme t_i est représenté par un vecteur de documents dans un espace de vecteur de documents $\vec{t_i} = (d_{i1}, ..d_{ij}, .., d_{in})$, où d_{ij} est le poids du terme t_i dans le document d_j :

$$d_{ik} = \frac{(0.5 + 05\frac{ff(d_k,t_i)}{maxff()t_i})iif(d_k)}{\sqrt{\sum_{j=1}^{j=n}((0.5 + 05\frac{ff(d_j,t_i)}{maxff()t_i})iif(d_j))^2}}$$

où :

$ff(d_k, t_i)$: représente la fréquence du terme t_i dans le document d_k,

$iif(d_k) = log(\frac{m}{d_k})$: représente la fréquence inverse en termes du document d_k ; avec m le nombre de termes dans la collection et $|d_k|$ est le nombre de termes dans le document d_k,

$maxff(t_i)$: représente la fréquence maximale du terme t_i dans toute la collection,

La construction du thésaurus se fait par le calcul de similarité entre les différents termes d'indexation. La similarité entre deux termes t_i et t_j est exprimée par le produit scalaire :

$$sim(t_i, t_j) = \vec{t_i}.\vec{t_i} = \sum_{k=1}^{n} d_{ik}.djk$$

Une fois le thésaurus construit, l'expansion d'une requête q donnée consiste à calculer la similarité $simqt(q, t)$ entre chaque terme d'indexation t et la requête q :

$$simqt(q,t) = \sum_{t_i \in q} q_i.sim(t_i,t)$$

Ainsi, tous les termes d'indexation peuvent être ordonnés par ordre décroissant de leurs valeurs de similarité avec la requête. Les k premiers termes les plus similaires à la requête sont alors rajoutés à la requête. Les poids des termes sélectionnés sont calculés comme suit :

$$w(q,t) = \frac{simqt(q,t)}{\sum_{t_i \in q} q_i}$$

Le nombre de termes à rajouter est un paramètre important. L'étude empirique effectuées dans [QF93] sur 3 collections documentaires (MED, CACM, NPL) a montré que le nombre de termes à rajouter doit avoisiner les 100. Cependant, d'une collection à l'autre le nombre optimal de termes à rajouter est très variable, ainsi les valeurs optimales pour MED, CACM et NPL sont respectivement 80, 100, 800. Ainsi si le nombre de termes rajoutés à la requête est insuffisant les performances sont médiocres. Inversement si le nombre de termes rajoutés à la requête est excessif les performances de la nouvelle requête peuvent être inférieure à la requête initiale.

2.5.3.2 Thésaurus statistique globale

Dans cette section nous présentons la technique d'expansion de requête basée sur un thésaurus appelé thésaurus statistique Global [CY92]. Le thésaurus statistique Global est composé de classes de termes extraites de la collection. L'expansion d'une requête consiste à rajouter parmi les classes de termes, la classe la plus similaire à la requête.

L'objectif recherché de toute méthode d'expansion de requêtes est de sélectionner des termes discriminants pour l'expansion de requêtes. Ainsi dans [CY92] les termes sélectionnés pour la construction des classes de termes sont ceux dont la fréquence en document est faible, car ce type de termes sont discriminants [SYY75]. Cependant il est difficile de classer ce type de termes car ils apparaissent dans peu de documents. Par conséquent, au lieu de classer les termes, les auteurs proposent de classer les documents et d'associer aux classes ainsi construites les termes dont la fréquence en documents est faible.

La classification des documents s'effectue de la manière suivante :

1. chaque document de la collection est associé à une classe distincte, i.e. le nombre de classes correspond au nombre de documents de la collection,

2. calculer la similarité entre chaque paire de classes,
3. déterminer la paire de classes (C_u, C_v) dont la valeur de similarité est la plus grande,
4. fusionner les classes C_u et C_v,
5. si le nombre de classes est supérieur à 1 alors allez à l'étape 2,
6. retourner la structure hiérarchique des classes de documents.

La similarité entre deux classes de documents correspond au minimum des similarités entre toutes les paires de documents, i.e. deux documents n'appartenant pas à la même classe. Les documents sont représentés par des vecteurs de termes et la similarité entre deux documents est calculée par le cosinus entre les vecteurs représentant les documents.

La classification ainsi effectuée produit une structure hiérarchique des documents de la collection.

la détermination des classes de termes s'effectue, selon trois paramètres :

- TC : seuil de similarité entre deux classes de documents, il permet de déterminer les classes de documents qui seront utilisés pour la génération des classes de termes, i.e. la paire de classess (C_u, C_v) sera fusionnée si la similarité $sim(C_u, C_v)$ entre C_u et C_v est supérieure ou égale à TC. Si $sim(C_u, C_v) < TC$ alors C_u et C_v seront considérés comme des classes distinctes
- NDC : nombre maximum de documents d'une classe. Il représente une condition supplémentaire pour la génération des classes de documents. Ainsi si le nombre de documents de la paire de classes (C_u, C_v) est supérieur à NDC alors C_u et C_v ne seront pas fusionnées même si la similarité $sim(C_u, C_v)$ entre C_u et C_v est supérieure ou égale à TC.
- $MIDF$: seuil de la fréquence inverse en documents.

Une fois les classes de documents construites (via les paramètres TC et NDC), les documents de chaque classe sont utilisés comme source de termes. Ainsi à chaque classe de documents on associe une classe de termes, i.e. les termes contenus dans les documents de la classe de document.

Afin de sélectionner des termes discriminants seuls les termes dont la fréquence en document est faible seront considérés, i.e. les termes dont la fréquence inverse en documents est inférieure à un seuil donnée $MIDF$.

Une fois les classes de termes construites, i.e. les classes du thésaurus, celle-ci peuvent être utilisées pour l'expansion de requêtes. L'expansion se fait par l'ajout

de tous les termes de la classe la plus similaire à la requête. Le poids des termes d'une classe C sont égaux et dont la valeur est donnée par la formule :

$$wt_C = \frac{\sum_{i=1}^{|C|} w_{i,C}}{|C|}$$

Les expériences effectuées sur 4 collections documentaires ont montré une amélioration significative des performances. Cependant le problème majeur de cette approche réside dans les paramètres TC, NDC et $MIDF$ dont les valeurs optimales diffèrent d'une collection à une autre.

2.6 Évaluation des systèmes de recherche d'information

Les systèmes de recherche d'information peuvent être évalués selon deux classes de critères :

- Les critères d'efficience, comme la manière dont les résultats sont présentés à l'utilisateur, la facilité d'utilisation du système, le temps d'attente, etc.
- Les critères de pertinence, tel que la précision des documents retrouvés, le rappel, etc.

Les critères d'efficience sont très subjectifs et dépendent fortement des jugements de l'utilisateur. Par conséquent, dans cette section nous nous focaliserons sur les critères de pertinence.

2.6.1 La pertinence des documents

Un document est dit pertinent vis à vis d'une requête si l'information qu'il apporte correspond aux besoins d'information exprimé par la requête. Cette pertinence est très subjectives car plusieurs personnes peuvent avoir des avis différents sur la pertinence d'un document vis à vis d'une requête. Cependant, des études ont montrées que les différences sont assez faibles [Rij79].

Généralement, l'évaluation des SRI s'effectue sur des collections de documents test ou pour chaque requête une liste de documents pertinents est produite par des experts.

2.6.2 Precision et Rappel

Ce sont les mesures principales pour l'évaluation des systèmes de recherche d'information.

$$Précision = \frac{nombre\,de\,documents\,pertinents\,retournés}{nombre\,total\,de\,documents\,retournés}$$

$$Rappel = \frac{nombre\,de\,documents\,pertinents\,retournés}{nombre\,total\,de\,documents\,pertinents}$$

Plus formellement, pour une collection de documents D et une requête de recherche donnée q, soit les ensembles $Pert$ et Ret (voir figure 2.4), tel que :

- $Pert$ représente l'ensemble de documents pertinents pour la requête q
- Ret représente l'ensemble de documents retournés par le système

Alors la précision notée P et le rappel noté R sont calculés comme suit :

$$P = \frac{|Ret \cap Pert|}{|Ret|}$$

$$R = \frac{|Ret \cap Pert|}{|Pert|}$$

Ainsi, la précision mesure le degré de pertinence des documents retournés par le système, i.e. sa capacité de retourner des documents pertinents et à rejeter les documents non pertinents. Quant au rappel, il mesure la capacité du système à retrouver tous les documents pertinents correspondants.

2.6.3 Courbe Rappel/Précison

Souvent les résultats de la recherche sont évaluées à l'aide d'une courbe de rappel/précision. Comme l'illustre la figure 2.5, la courbe de rappel/précision est représenté par un ensemble de points (R_i, P_i), où pour chaque valeur de rappel R_i est associée une valeur de précision P_i.

Ce graphe est généralement obtenu en interpolant les valeurs de précision associées aux différentes valeurs de rappel. L'interpolation est généralement effectuée en choisissant pour un niveau de rappel P_i donné une précision égale à la précision maximale obtenue pour tout rappel supérieur à P_i [VH99]. [2]

[2] cette règle est celle utilisée par l'outil trec_eval dans le cadre des compagnes d'évaluation TREC et AMARYLLIS.

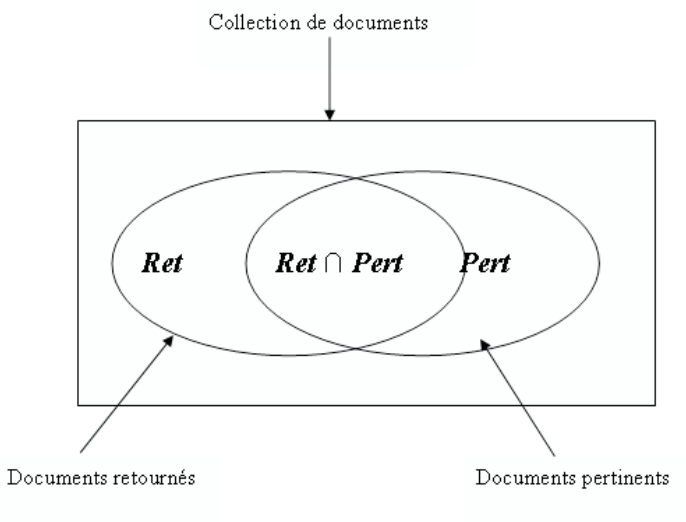

FIG. 2.4 – Illustration des ensembles utilisés dans le calcul de la précision et du rappel

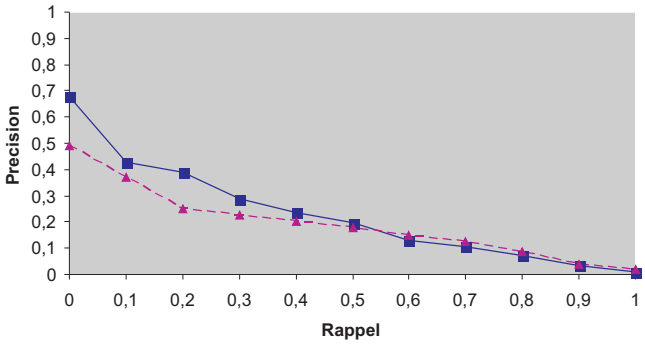

FIG. 2.5 – exemples de courbes de précision/rappel

La courbe de précision/rappel d'un système parfait, i.e. celui qui trouverait seulement les documents pertinents, serait une droite horizontale à 1. Lorsqu'on compare deux courbes de précision/rappel, la meilleure sera celle au dessus de l'autre. La comparaison devient plus délicate pour des courbes qui se croisent.

2.6.4 Autres mesures

Les mesures de rappel et de précision sont les plus utilisées, cependant différentes autres mesures ont également été proposé [SG83][JBH97][Rij79][TS92] [Bes02] :

- la R-précision : il s'agit de la précision obtenue pour un nombre de documents retournés correspondant au nombre de documents pertinents.
- la longueur de recherche : elle correspond au nombre de documents non pertinents que doit lire l'utilisateur pour avoir un certain nombre n de documents pertinents.
- la mesure d'efficacité "E" : est une combinaison des valeurs de rappel et de précision :

$$E = 1 - \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}}$$

  Cette mesure permet à l'utilisateur de spécifier laquelle parmi les valeurs de précision et de rappel est plus intéressante, ainsi :
  - quand $\alpha \to 1$, l'utilisateur ne s'intéresse pas à la précision
  - quand $\alpha \to 0$, l'utilisateur ne s'intéresse pas au rappel.
  - quand $\alpha = 0.5$, l'utilisateur donne autant d'importance à la précision et au rappel

  Notons que cette mesure est une mesure inverse, i.e. une petite valeur de E indique une grande efficacité
- La mesure harmonique : elle correspond à la moyenne harmonique de la précision et du rappel

$$H = \frac{2PR}{P+R}$$

  La valeur de H est comprise entre 0 et 1, plus H s'approche de 1 plus le système évalué est efficace. Notons que la mesure harmonique H correspond à $1 - E$ pour $\alpha = 0.5$

2.7 Conclusion

Ce premier chapitre à porté essentiellement sur l'étude des systèmes classiques de recherche d'information. Nous avons ainsi passé en revue les différentes techniques de sélection et de pondération des termes d'indexation, les différents modèles de RI et les méthodes de pondération des termes.

Ces techniques permettent de résoudre quelques problèmes inhérent à la recherche d'information. Cependant, quelque soit la puissance du SRI utilisé quelques problèmes persistent :

1. la requête de recherche est souvent exprimées par une liste de termes souvent très réduite qui exprime souvent mal les besoins en information de l'utilisateur, ainsi des documents pertinents sont souvent omis ou mal placés. En effet même si des mécanismes d'enrichissement automatique de requête permettent d'améliorer la qualité des documents retournés, cependant cette amélioration reste faible,
2. pour une même requête le système retourne les même résultats, alors que des utilisateurs différents peuvent formuler la même requête pour exprimé des besoins différents. Également un même utilisateurs peut utiliser la même requête sur des périodes différents pour exprimer des besoins diffèrent,
3. Enfin, les systèmes de recherche d'information nécessitent de formuler de nouvelles requêtes à chaque nouveaux besoins d'informations.

Une solution pouvant pallier ces problèmes consiste à tenir compte du profil de l'utilisateur pour : personnaliser sa recherche ou lui fournir des information de manière automatique (filtrage et recommandation). Ce processus de prise en compte du profil utilisateur pour la fourniture d'information est appelé personnalisation.

Nous présentons dans le prochain chapitre les principaux concepts et techniques pour l'accès personnalisé à l'information.

Chapitre 3

Personnalisation de l'information

Sommaire

3.1 Introduction

Avec le développement d'Internet et des supports de stockage, la quantité de documents disponibles ne cesse de croître. Trouver l'information pertinente adaptés aux besoins de l'utilisateur est une tache délicate. Cette situation a introduit le besoin de concevoir des outils permettant un accès personnalisé à l'information. La personnalisation de l'information est actuellement en plein essor et touche plusieurs domaines tel que : la recherche d'information, la recommandation d'information, les interfaces adaptatrices, le filtrage d'information, les agents personnels, la veille technologique, etc.

Quelle que soit l'approche de personnalisation, celle-ci s'effectue à travers la notion de profil utilisateur. Ce dernier représente une certaine connaissance acquise sur l'utilisateur.

Dans ce chapitre nous nous intéressons a la personnalisation de documents textuels. Ainsi dans la section 3.2 nous aborderons la notion de profil utilisateur de manières générale. Dans la section suivante nous présentons les différents classes de personnalisation d'information à savoir : les systèmes de recherche personnalisé d'information et les systèmes de collecte passive d'information. nous décrivons pour chaque classes nous décrivons les approches les plus significatives. Enfin nous terminons ce chapitre par

3.2 Le profil utilisateur

Quelle que soit l'approche de personnalisation, on a toujours besoin de collecter et sauvegarder des données décrivant les utilisateurs sous forme de profils.

Le profil utilisateur peut être définit comme étant de la connaissance acquise sur l'utilisateur, cette connaissance peut correspondre à ce que l'utilisateur préfère, à l'historique des actions de l'utilisateur et leur évolution dans le temps, .etc.

3.2.1 Acquisition du profil

Il y a plusieurs façons d'obtenir ces caractéristiques en fonction du degré d'autonomie du système, de ses capacités d'observation et d'adaptation.

– Profils statiques d'experts : cette méthode nécessite que l'on dresse la liste des profils et que l'on décrive chacun d'entre-eux. Cette approche est statique,

donc une fois que le système a démarré, il est difficile de mettre à jour les profils utilisateurs.
- Profils reflexifs : L'utilisateur doit remplir des formulaires pour configurer son propre profil. Cette approche permet plus de précision et d'adaptation.
- Profils corrigés dynamiquement : Un sous-système de modélisation observe l'utilisateur de derrière l'interface et apprend le profil de l'utilisateur à partir de ses actions.

Les deux premières méthodes d'acquisition des profils sont statiques et ne permettent pas de s'adapter aux changements des centres d'intérêts de l'utilisateur. Par conséquent l'acquisition de profils corrigés dynamiquement est la plus adéquate.

Dans ce cadre, le retour de pertinence utilisateur peut se faire de manière explicite [DN02, CS98, Moc96] et/ou implicite [Bal97a, Bal97b, TT98, GCP03]. Dans le cas d'un retour de pertinence explicite, l'utilisateur indique au système les informations qu'il considère pertinentes ou non. Quant au retour de pertinence implicite, c'est le système qui déduit au travers les actions de l'utilisateur de la pertinence ou non des informations manipulées par l'utilisateur.

Le retour de pertinence peut être positif [MS94, WPW95b, MWB94, SBR02], positif et négatif[WIY01, WIY03, She94, MZ97], flou [DN02]. Ainsi, un retour de pertinence positif (resp. négatif) indique que l'utilisateur a trouvé l'information délivré pertinente (resp. non pertinente). Un retour de pertinence flou, consiste à donner un score de pertinence à l'information délivrée.

3.2.2 Apprentissage des profils

Différentes techniques d'apprentissage du profil ont été proposées dans la littérature, la majorité d'entre elles modélisent le profil utilisateur par des vecteurs de termes et cela en utilisant des versions incrémentales de l'algorithme de Rocchio [Roc71], ainsi on y trouve les travaux de [All96, WIY01, DN02]. Cependant d'autres techniques d'apprentissage ont été proposées dans la littérature, ainsi on y trouve des techniques d'apprentissage basés sur les classifieurs Bayésiens [PB97, KHZ00], les réseaux de neurones [MS94, Che04], les algorithmes génétiques [She94, MZ97, MWB94],..etc.

Nous présentons dans les sous sections suivantes quelques techniques d'apprentissage les plus répandues, à savoir : les approches vectorielles, les réseaux de neurones et celles basés sur les algorithmes génétiques.

	Retour de pertinence utilisateur				
	Positif	Négatif	Flou	Explicite	Implicite
Danilowicz & all			•	•	
WebMate	•			•	
McEligot & all	•			•	
Wiener & all	•			•	
Menczer & all	•	•		•	
INFOS	•	•		•	
AIRA	•		•		•
Amalthea	•	•		•	
NewT	•	•		•	
FAB	•	•		•	•
PIN	•	•		•	•
ALIPES	•	•		•	

TAB. 3.1 – Caractéristiques du retour de pertinence utilisateur de quelques systèmes de personnalisation

3.2.2.1 Approches vectorielles

La représentation des profils utilisateurs par des vecteurs de termes est l'approche la plus utilisée dans la littérature [WIY01, DN02, CS98, MK90]. Elle est basée sur le principe de re-injecion de pertinence introduit par Rocchio [Roc71].

Dans *Fab* [Bal97a, Bal97b], un système de recommandation de pages Web, le profil utilisateur est représenté par un vecteur de termes, pondérés selon TF-IDF. La mise à jour du vecteur profil s'effectue au travers les jugements de l'utilisateur sur la pertinence ou non des pages Web proposées.

Dans le modèle de recherche d'information proposé dans [DN01, DN02], Le profil utilisateur est modélisé par un vecteur de termes dans le même espace vectoriel que les documents et les requêtes. A chaque requête de recherche effectuée par l'utilisateur, son profil est mis à jour au travers ses jugements sur les documents retournés.

Dans l'Agent de filtrage *Alipes* [WIY99, WIY01], le profil utilisateur est représenté par plusieurs vecteurs. Les documents jugés par l'utilisateur sont catégorisés et à

chaque catégorie c_i est associée trois vecteurs $POSD_{c_i}$, $NEGD_{c_i}$ et LTV_{c_i}.
$POSD_{c_i}$ (respectivement $NEGD_{c_i}$) est construit à partir des documents de la catégorie jugés pertinents (respectivement non pertinents) par l'utilisateur.
LTV_{c_i} est le vecteur représentant l'intérêt long terme de l'utilisateur pour la catégorie c_i. Il est construit à partir des documents de la catégorie jugés pertinents et non pertinents par l'utilisateur. Le filtrage de documents se fait par la calcul de similarité entre les vecteurs représentant les documents et le vecteur TDR qui est une combinaison des vecteurs $POSD_{c_i}$, $NEGD_{c_i}$ et LTV_{c_i}.

3.2.2.2 Réseaux de neurones

Plusieurs systèmes de personnalisation utilisant les réseaux de neurones pour l'apprentissage du profil utilisateur ont été proposés [WPW95a, TT98, MS94, JH92, Che04].

Dans les système de filtrage d'articles sur Internet [MS94], un réseaux de neurones en couches est utilisé pour la modélisation du profil utilisateur. Une couche d'entrée recevant les documents à filtrer et une couche de sortie pour les documents susceptibles d'intéresser l'utilisateur. Le système fonctionne en deux modes : le mode apprentissage et le mode comparaison. En mode apprentissage, des documents correspondant aux intérêts de l'utilisateur sont soumis au système pour l'apprentissage du réseau. En mode comparaison, les documents collectés par le système sur le Web sont filtrés à l'utilisateur. Le système bascule ensuite vers le mode apprentissage pour recevoir les jugements de l'utilisateur sur les articles filtrés dans la phase précédente.

Dans le système de recommandation de documents proposé dans [TT98], un réseau de neurones appelé ARAM (Adaptative Resonance Assoicative Map) est utilisé pour l'apprentissage du profil utilisateur. ARAM est un réseau à apprentissage par compétition de la famille ART (Adaptive Resonance Theory) [CGR91]. ARAM organise les informations en catégories et cela en fonction de leurs similarités. Le profile utilisateur est modelisé en lui associant des catégories et cela en fonction de ces retour de pertinence.

3.2.2.3 Approches basés sur les algorithmes génétiques

Les algorithmes génétiques sont inspirés du concept de sélection naturelle élaboré par Charles Darwin. Il a montré que l'apparition d'espèces distinctes se fait par le biais de la sélection naturelle fondée sur la lutte pour la vie, due à une population tendant naturellement à s'étendre mais disposant d'un espace et de ressources finis.

Plusieurs approches utilisant les algorithmes génétiques pour l'apprentissage des intérêts de l'utilisateurs ont été proposés [She94, MZ97, MWB94]. La première application utilisant les algorithmes génétiques pour la modélisation du profil utilisateur a été proposé par Sheth dans son agent *NewT*, dédié à la collecte et au filtrage d'information sur le Web [She94]. *NewT* contient une population de profils. A chaque profil est associé un poids désignant son adaptabilité et chacun des profils collecte et filtre des documents à l'utilisateur. Les documents filtrés sont évalués par l'utilisateur, ainsi, plus un profil propose des documents pertinents plus sa comptabilité augmente. A chaque cycle de filtrage, les profils dont le poids d'adaptabilité est faible sont éliminés et remplacés par des profils provenant de combinaison génétiques de ceux dont les poids d'adaptabilité sont importants.

3.3 Les systèmes de personnalisation d'information

Avec le développement d'Internet la quantité de l'information disponible ne cesse d'augmenter. L'utilisateur finale est submergé d'informations et il n'est plus capable d'appréhender ces informations de manière efficace. Cette situation à introduit le besoin d'adapter l'information aux besoins de l'utilisateur. Ainsi de plus en plus de systèmes d'information proposent des services de personnalisation afin de mieux cibler leur clients.

Les systèmes de personnalisation les plus couramment proposés peuvent être classés en deux catégories :

- Les systèmes de personnalisation de la recherche d'information : permettent de fournir à l'utilisateur des informations personnalisées en réponse à un besoin d'information exprimé par une requête de recherche et cela en tenant compte de son profil.
- Les systèmes de recommandation : permettent de fournir à l'utilisateur de manière automatique et sans formulation de requêtes des informations adaptés aà ses besoins. Cette tache s'effectue en analysant ses préférences (son profil) et/ou en se servant de l'expérience des autres utilisateurs.

Dans les sous sections suivantes nous décrivons de manière détaillée chaque catégorie ainsi que quelques approches proposées dans la littérature.

3.3.1 Systèmes personnalisés de recherche d'information

La recherche d'information s'effectue au travers une requête de recherche. Cependant cette dernière traduit souvent mal les besoins en information de l'utilisateur. D'une part, celle-ci est souvent exprimée par une liste de termes souvent très réduite qui exprime mal les besoins en information de l'utilisateur. D'autres part deux utilisateurs peuvent formuler la même requête pour exprimer des besoins différents.

La personnalisation de la recherche d'information peut se faire selon deux stratégies :

- Adaptation au profil utilisateur des résultats de la requête initiale : consiste à utiliser les résultats de la requête initiale pour les adapter au profil de l'utilisateur. Cette adaptation des résultats peut se faire par ré-ordonnancement [BGB03, PG99, GCP03] ou par filtrage [BRS00, SBR02, RBS00]. Le ré-ordonnancement consiste à modifier l'ordre d'affichage des résultats en analysant le profil utilisateur. Le filtrage des résultats consiste à sélectionner parmi les résultats de la requête initiale ceux susceptible d'intéresser l'utilisateur.
  L'avantage du filtrage et du re-ordonnancement des résultats est leur simplicité car elles ne nécessitent aucune modification du système de recherche d'information, ainsi le traitement est fait après l'exécution de la requête.
- Intégration du profil utilisateur dans le processus de recherche : contrairement à la premières approche, celle-ci utilise le profil utilisateur pour explorer de nouveaux espaces de recherche. Cette exploration peut se faire en étendant la requête au travers l'analyse du profil utilisateur [Bot04, MK90, BH00, CK00, LYM04] (i.e. sans changement majeur du fonctionnement du système de recherche) ou par des approches dont le fonctionnement intègre le profil utilisateur [DN01, DN02].

3.3.1.1 Adaptation au profil utilisateur des résultats de la requête initiale

3.3.1.1.1 Le ré-ordonnancement des résultats Consiste à effectuer une requête de recherche sans tenir compte du profil de l'utilisateur. Ensuite un post traitement est effectué sur les résultats afin d'adapter l'ordre d'affichage des résultats aux préférences de l'utilisateur.

Différentes approches de ré-ordonnancement des résultats de la recherche ont été proposées [BGB03, PG99, GCP03].

Dans l'approche proposé dans [PG99, GCP03], le profil de est construit au tra-

vers l'analyse des documents consultés par l'utilisateur (pages Web). Le profil est représenté par une hiérarchie de catégories où à chaque catégorie est associé un ou plusieurs documents. Cette hiérarchie est obtenue par la sélection dans une ontologie générale[1] de noeuds, ceux estimés correspondants aux intérêts de l'utilisateur.

La nouvelle position $p(d_i)$ d'un document d_i est en fonction de :

- l'ancienne position $p_0(d_i)$ attribué par le moteur de recherche,
- des quatre catégories c_1, c_2, c_3, c_4 auquels l'utilisateur s'intéresse le plus et de son intérêt $\pi(c_j)$ ($j = 1$ à 4) pour chacune de ces catégories,
- de la similarité $\gamma(c_j, d_i)$ ($j = 1$ à 4) entre le document d_i et chacune des catégories auquel l'utilisateur s'intéresse le plus.

Ainsi nous avons :

$$p(d_i) = p_0(d_i) * [0.5 + 0.25 \sum_{j=1}^{j=4} (\pi(c_j) * \gamma(c_j, d_i))]$$

où :

temps d : représente le temps passé par l'utilisateur pour la lecture du document d

longueur d : représente la taille en nombre de caractères du document d,

Les expérimentations effectués sur 16 utilisateurs effectuant chacun d'eux 3 requêtes de recherche ont montrés une amélioration de la précision des résultats par rapport à ceux fournis directement par le moteur de recherche. Cependant cette amélioration reste peu significative.

3.3.1.1.2 Le Filtrage des résultats Consiste à effectuer une requête de recherche sans tenir compte des préférences de l'utilisateur et d'effectuer ensuite un post traitement sur les résultats afin d'éliminer les résultats non pertinents pour l'utilisateur et cela par l'analyse de son profil.

Un exemple de système de filtrage de résultats est CASPER, dédié à la recherche d'annonce de travail en ligne [BRS00, SBR02, RBS00]. Dans celui-ci le traitement d'une requête de recherche s'effectue en deux phase :

1. d'abord un calcul de similarité entre la requête et la base de données est effectué pour extraire un certain nombre d'annonces. Ainsi les annonces dont la similarité avec la requête est supérieure à un seuil donnés sont sélectionnés.

[1] Magellan : hiérarchie contenant approximativement 4400 nuds (magellan.excite.com)

Notons que l'objectif de l'appariement approché entre la requête et les annonces est l'élargissement du champs de recherche. Ainsi dans cette première phase des annonces ne correspondant pas tout à fait aux besoins d'information exprimés aà travers la requête de recherche sont sélectionnés.

2. les résultats de la première phase sont ensuite comparés au profil utilisateur. Ainsi seul les annonces proches des centres d'intérêts de l'utilisateur sont proposées à l'utilisateur.

Les termes des annonces sont représentés par des arbres ontologiques et la similarité entre deux termes s'effectue par le calcul de la distance entre eux dans l'arbre. Ainsi la similarité entre la requête de recherche et une annonce donnés s'effectue en comparant les attributs de la requêtes et ceux de l'annonce.

Le profil utilisateur est composés de deux classes d'annonces de travail celles jugés pertinente et celle jugés non pertinent par l'utilisateur. Ces jugements sont effectués sur les résultats des précédentes recherche. La similarité entre une annonce est un profil est une combinaison des similarités entre l'annonce et l'ensemble des annonces contenues dans le profil. La similarité entre deux annonces est en fonction des similarités entre les attributs de ces annonces.

3.3.1.2 Intégration du profil utilisateur dans le processus de recherche

3.3.1.2.1 L'enrichissement des requêtes L'enrichissement d'une requête consiste en l'ajout et/ou retrait de termes de la requête initiale [Eft00]. Dans le cadre de la personnalisation de la recherche d'information l'enrichissement des requêtes s'effectue au travers l'analyse du profil utilisateur afin d'adapter la nouvelle requête au profil de l'utilisateur.

Différentes approches ont été proposé dans la littérature [Bot04, MK90, BH00, CK00, LYM04]. Par exemple, dans AIRA [Bot04], un système de recherche personnalisé d'information sur le Web par enrichissement de requête. Le profil utilisateur est modélisé par un profil long terme et un profil court terme. Le profil long est construit de manière statique, il est représenté par une hiérarchie de termes construit à partir d'un ensemble de documents ou de références sélectionnés par l'utilisateur. Le profil court terme est construit dynamiquement, il contient les dernieres informations textuels manipulé par l'utilisateur (pages Web visualisées, contenu du presse-papier, documents en cours d'édition, etc.).

L'expansion de la requête s'effectue en deux phases :

1. Une phase de détermination du vocabulaire : elle consiste à identifier les termes les plus efficaces permettant d'étendre la requête initiale. Les termes sélectionnés sont ceux contenus dans la requête de recherche, les termes provenant de l'activité courante de l'utilisateur (profil court terme, c'est à dire les derniers documents manipulés) ainsi qu'un ensemble de termes sélectionnés parmi ceux du profil long-terme. Le choix de ces derniers dépend de l'activité courante de l'utilisateur. Ainsi les noeuds sélectionnés sont ceux dont les termes sont contenus dans les documents de l'activité courante de l'utilisateur.
2. Une phase d'apprentissage de la requête : elle consiste à rechercher parmi les termes sélectionnés lors de la premières phases, la combinaison de termes optimale, c'est à dire celle permettant d'obtenir les document ayant la meilleur évaluation de pertinence. Ainsi un ensemble de requêtes est créé en combinant la requête initiale soumise par l'utilisateur avec des termes tirés au hasard de l'ensemble des termes du vocabulaire. Ces requêtes sont ensuite soumises aux moteur de recherche. Pour chaque requête, une évaluation de pertinence est calculée pour les documents retournés par le moteur, par rapport à la requête utilisateur et a son contexte. A partir des requêtes les mieux évalués, de nouvelles requêtes sont générés et évalués. Ce processus est répété jusqu'a ce qu'on atteint le nombre maximum de cycles, ou si aucun documents n'est retournés. Enfin à la fin de cette phase les meilleurs documents obtenus sont présentés à l'utilisateur, classés par ordre d'évaluation de pertinence.

Les résultats empiriques montrent une amélioration significative des résultats par rapport a ceux des requêtes initiales. Cependant le caractère statique du profil utilisateur est un handicape majeur car l'utilisateur doit mettre à jour manuellement son profil long terme. Il est a noter aussi que les résultats risque d'être peux pertinents si l'activité courante n'est pas en relation avec le besoin d'information de l'utilisateur exprimé par la requête.

3.3.1.2.2 Architectures basées sur le profil Le système *AIRA* cité précédemment intègre le profil utilisateur dans le processus de recherche par l'ajout de termes à la requête initiale, cependant le fonctionnement du système de recherche utilisé n'est point modifié. D'autres systèmes dont l'architecture dépend du profil utilisateur ont été proposés [DN01, DN02].

Par exemple dans le modèle proposé dans [DN01, DN02], le profil utilisateur est représenté par un vecteur de termes. La construction du profil utilisateur s'ef-

fectue par l'addition linéaire des vecteurs de documents consultés par l'utilisateur. Le modèle de recherche d'information proposé est représenté par le quintuple $\langle D, Q, P, s, f\rangle$, où D représente l'ensemble des documents de la collection, Q l'ensemble des requêtes, P l'ensemble des profils utilisateur, s est une mesure de similarité et f une fonction de recherche.

Pour un document d, une requête q et un profil p donnés, la fonction de recherche $f(q, p, d)$ est défini par :

$$f(q,p,d) = \alpha_1.s(q,d) + \alpha_2.\sqrt[\beta]{(s(q,d))^{\beta_1}.(s(p,d))^{\beta_2}}$$

où $\alpha_1, \alpha_2, \beta_1, \beta_2, \beta$ sont des constantes tel-que :
$\alpha_1 > 0$ et $\alpha_2, \beta_1, \beta_2 \geq 0$ et $\beta = \beta_1 + \beta_2$

Le profil utilisateur est modélisé par un vecteur de termes dans le même espace vectoriel que les documents et les requêtes. Sa création et sa mise à jour est basée sur les jugements utilisateur des documents retrouvés lors des précédentes recherche. Soit t le nombre courant indiquant le nombre de modification effectuées sur le profil, le processus de mise à jour peut se résumer comme suit :

1. l'utilisateur soumet une requête q au système de recherche
2. si $t = 0$ (i.e. le profil est vide), le système effectue la recherche en se basant sur la similarité $s(q, d_i)$. si $t \neq 0$ Alors la recherche de documents est basé sur la fonction $f(q, p, d_i)$.
3. soit $o^t = (d_1^t, d_2^t, .., d_n^t)$ l'ordre des documents retournés retournés par le système pour la requête q. l'utilisateur juge les documents retournés par le système, il s'effectue en ré-ordonnant les documents proposés par le système selon ses préférences $o^u = (d_1^u, d_2^u, .., d_n^u)$
4. un profil temporaire p^u est crée en utilisant l'ordre o^u :

$$p^u = \frac{\sum_{i=1}^{n} w_i * d_i^u}{\sum_{i=1}^{n} w_i *}$$

   $w_1, .., w_n$ sont des paramètres indiquant le poids d'un document dans le profil p^u (pour le détail sur le calcul de ces paramètres voir [Dan94]).
5. le profil p est mis à jour comme suit :
   - si $t = 0$ alors le profile p^1 est crée, avec $p = p^u$
   - si $t > 0$ alors le profil est mis à jours comme suit :

$$p^{(t+1)} = \frac{p^t * t + p^u}{t+1}$$

Le modèle n'as pas été évalué empiriquement. Cependant nous pouvons constater que le profil qui est une addition linéaire des vecteurs de documents jugés pertinents par l'utilisateur peut devenir très générique. Par conséquent la précision des documents retournés peut être sensiblement altéré.

3.3.2 Les systèmes de collecte passive d'information

La collecte passive d'information consiste à proposer àl'utilisateur de manière automatiques des information adaptés à ses besoins. Cette tache s'effectue en analysant ses préférence (son profil) ou en se servant de l'expérience des autres utilisateurs.

Une multitudes de terminologies différentes sont utilisés pour décrire les différentes formes de collecte passive d'information, tels que : filtrage, routage, recommandation, diffusion sélective d'information. Par conséquent, nous utiliserons le terme "filtrage d'information" pour designer la sélection d'information exclusivement adaptées a l'utilisateur et provenant de sources dynamique et le terme "recommandation" pour les autres formes de collecte passive d'information.

3.3.2.1 Le Filtrage d'information

Le filtrage d'information (FI)est un processus dont l'objectif est de faire parvenir à partir d'un ensemble d'informations provenant de sources dynamiques, celles susceptibles de correspondre au intérêts de l'utilisateur.
Les domaines d'application du filtrage sont très diverses, parmi eux : mailing list, filtrages des e-mails, veille technologique, ..etc.

Comme l'illustre la figure 3.1, un système de filtrage d'information (SFI) est composé d'un flot d'informations (*Web*, *Emails*, *News*, etc.), d'utilisateurs ayant chacun un profil et de fonctions de décision. A chaque arrivée d'un document, la décision quant à son acheminement ou non vers un utilisateur est assurés par la fonction de décision associée au profil de ce dernier. Cette tache s'effectue en comparant le document au profil utilisateur et/ou en le comparant avec les autres profils.

3.3.2.1.1 Les différentes formes de filtrage Le filtrage d'information peut être effectués de plusieurs manières, dans [MGT+87] l'auteur a identifié trois formes de filtrages :

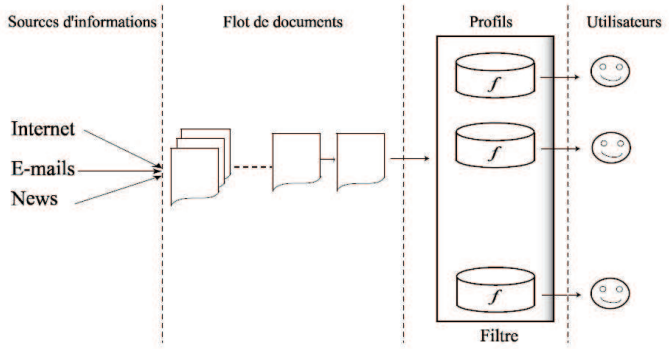

FIG. 3.1 – Architecture fonctionnelle d'un système de filtrage d'information

1. **le filtrage social** : appelée aussi filtrage collaboratif, il se base sur le travail collaboratif entre différents utilisateurs [GNOT92, SKKR01, Lee01, CLP02, MH04]. L'objectif est de faire profiter l'utilisateur de l'expérience des autres utilisateurs. Ainsi, la décision de la sélection ou non d'un document est basée sur les jugements et les annotations effectuées par d'autres utilisateurs sur ces documents. Le filtrage de documents peut être basé uniquement sur les liens entre les utilisateurs [HKBR99] ou basé sur le contenu des documents et les liens entre les utilisateurs (collaboration) [CLP02].
2. **Le filtrage économique** : dans ce type de filtrage la décision de sélection ou non d'un document se base sur le coût et l'intérêt du document. L'objectif est de filtrer les documents afin de minimiser les coûts et de maximiser les intérêts. Ainsi dans ce type de filtrage les profils utilisateurs intègrent des informations sur leurs préférences mais aussi des paramètres de coûts.
3. **le filtrage cognitif** : c'est le type de filtrage le plus répandu. La décision de sélection ou non d'un document dépend uniquement du contenu de l'information a filtrer [BT01, Teb04]. En d'autre termes la décision de sélection ou non d'un document se fait par la comparaison du contenu du document au profil de l'utilisateur.

3.3.2.1.2 Filtrage et recherche d'information Les systèmes de filtrage d'information sont souvent basés sur les techniques de RI. En effet comme le montre Belkin dans [BC92] le filtrage d'information et la recherche d'information sont des processus dual. Leur objectif commun est de sélectionner les informations répondant

au besoins d'information de l'utilisateur. Cependant dans un SFI [Bou00] :

- les documents arrivent continuellement de sources évolutives alors que dans un SRI les document proviennent d'une collection statique
- les besoins en information sont exprimé par un profil utilisateur,
- le processus de décision quant à la pertinence ou non d'un document est binaire alors que dans un SRI le processus est nuancé car il détermine un degré de pertinence.

La majorités des techniques d'apprentissage du profil utilisés sont des versions incrémentales de l'algorithme de Rocchio [Roc71], ainsi on y trouve les travaux de [All96, Cal98, SSS98, HMIH00, WIY01]. Cependant d'autres techniques d'apprentissage ont été proposés dans la littérature, ainsi on y trouve des techniques d'apprentissage basés sur les classifieurs Bayésiens [PB97, KHZ00] , les réseaux de neurones [KGDC00], les algorithmes génétiques [She94, MZ97],..etc.

3.3.2.2 Systèmes de recommandation d'information

La recommandation consiste à proposer à l'utilisateur de manière automatique des information adaptés à son profil ou en se servant de l'expérience des autres utilisateurs. La distinction entre le filtrage d'information et la recommandation n'est souvent pas claire, cependant dans un SFI [Bou00] :

- les sources d'informations sont souvent statiques, alors que dans un SFI les informations proviennent de sources dynamiques,
- le processus de décision quant à la pertinence ou non d'un document peut être binaire ou déterminé par un degré de pertinence.

Différentes applications de recommandation d'information ont été proposés dans la littérature, on peut citer : la recommandation de pages Web [MTP01], la recommandation de documents dans un espace d'information vaste (bibliothèques électroniques)[AMD+04], etc.

Par exemple dans [MTP01], un système de recommandation de pages Web par l'analyse des parcours utilisateurs a été proposés,.....A compléter par PASCAL PONCELET (merci d'avance).....

Une autre approche pour la recommandation de pages Web a été proposés par *Mobasher* dans [MDLN02, Mob]. La recommandation est basés sur l'analyse des comportements passés des autre utilisateurs. Ainsi à travers les traces de navigation (fichiers logs) des utilisateurs précédemment connectés le système construit des profils d'usage. Le système essaye de prédire les besoins d'un utilisateur connecté en

comparant son profil(les pages consultés) au profils d'usages.

1. construction des profils d'usage
   soit $P = \{p_1, p_2, .., p_n\}$ l'ensemble des pagesviews (vues sur les pages) contenues dans le serveur. Soit $T = \{t_1, .., t_m\}$ les transaction (traces d'usages) de m utilisateur.
   Chaque transaction t_i est représenté par un vecteur n-dimensionnels de page-views :

   $$t = \langle w(p_1, t), w(p_2, t), ..w(p_n, t)\rangle$$

   où $w(p_i, t)$ est le poids associé à la transaction t pour la pageview p_i.
   Les transactions sont ensuite catégorisées $TC = \{c_1, c_2, ..c_k\}$, où chaque catégorie c_i est un sous ensemble de l'ensemble des transaction T. La catégorisation est effectués de manière à maximiser la similarité entre les transactions de la même classe et à minimiser la similarité entre les différentes classes.
   pour chaque classe c est associé un profil d'usage comme suit :

   $$pr_c = \{\langle p, poids(p, pr_c)|p = pageview, poids(p, pr_c) \geq \mu\rangle\}$$

   où μ est un seuil donné,
   le poids de la pageview p vis à vis du profil d'usage pr_c est donné par :

   $$poids(p, pr_c) = \frac{1}{|c|}.\sum_{t \in c} w(p, t)$$

   où $w(p, t)$ est le poids de la pageview p pour la transaction t. chaque profil d'usage est ainsi représenté par un vecteur à n dimensions.
2. recommandation de pages Le système recommande des pages au utilisateurs connectés en comparant leurs profils (leur traces d'usages actuel) au profils d'usage.
   Ainsi soit $S = \langle s_1, s_2, .., s_n\rangle$ la trace d'usage d'un utilisateur connecté à un instant donné. tel que $s_i = 1$ si l'utilisateur à consulté la page p_i et 0 sinon.
   soit $pr_c = \langle w_1^c, w_2^c, .., w_n^c\rangle$ la représentation vectorielle du profil d'usage pr_c
   La trace d'usage de l'utilisateur est comparé avec l'ensemble des profils d'usage :

   $$match(S, pr_c) = \frac{\sum_{k=1}^{k=n} w_k * s_k}{\sqrt{\sum_{k=1}^{k=n}(s_k)^2 * \sum_{k=1}^{k=n}(w_k)^2}}$$

Un score de recommandation $Rec(S, p, pr_c)$ de chaque pageviewpour chaque profile d'usage pr_c est calculé de la maniere suivante :

$$Rec(S, p, pr_c) = \sqrt{weight(p, pr_c) * match(S, pr_c)}$$

ainsi le pagesviews dont $Rec(S, p, pr_c) >\geq \rho$ sont recomandés à l'utilisateur.

3.4 Conclusion

Ce chapitre à porté essentiellement sur l'étude des systèmes de personnalisation d'information d'une manière générale et plus particulièrement les systèmes de personnalisation de la recherche d'information et les systèmes de collecte passive d'information.

Différentes applications de personnalisation ont été proposés dan la littérature, cependant le socle commun de ces systèmes est la prise en compte du profil de l'utilisateur. Ainsi, l'efficacité d'un système dépend de la méthode d'apprentissage des profils ainsi que de la manière dont ces derniers sont intégré dans le processus de fourniture de l'information.

Dans les chapitres suivants nous présentons nos contributions : ainsi dans le chapitre 3 nous présentons le modèle PDIIR, une approche pour la recherche personnalisé d'information. Nous présentons dans le chapitre 4 une approche pour la recommandation d'information.

Chapitre 4

Le modèle PDIIR : une approche basée sur le profil utilisateur pour la recherche documentaire

Sommaire

4.1 Introduction

Avec le développement des supports de stockage, la quantité de documents disponibles est telle qu'il devient difficile de trouver l'information utile ou souhaitée par un utilisateur. Ce dernier généralement surinformé n'est plus capable d'appréhender ces informations de manière efficace.

Les systèmes classiques de recherche d'information permettent de retourner un ensemble de documents et cela aux travers de requêtes, souvent exprimés en langage naturel.

Malheureusement, quelque soit l'efficacité du système utilisé certains documents pertinents sont souvent omis ou mal placés car la requête de utilisateur exprime mal ses besoins en information. Blair et Maron [BM85] avaient montré que la faible performance des systèmes de recherche d'information est dû à l'incapacité des utilisateurs de formuler les requêtes adéquates.

La reponse du systeme de recherche Les besoins en information de l'utilisateur dépendent bien entendu de la requête exprimée mais également de la connaissance acquise par l'utilisateur dans le domaine de sa recherche : deux utilisateurs peuvent formuler les mêmes requêtes pour des besoins totalement différents. Par exemple, les résultats attendus pour un expert en langage java qui recherche de la documentation en formulant la requête "cours java" sont évidement différents des résultats attendus par un amateur qui formule la même requête.

One, solution consists of taking into account the user profile in the information retrieval process in order to increase the relevance of the answered documents.

4.2 Le Modèle PDIIR

L'objectif d'un système de recherche documentaire est de répondre par des documents pertinents à une requête souvent exprimée en langage naturel. Afin de répondre de manière plus adaptée, dans *DIIR* les termes d'indexation sont sélectionnés de manière dynamique. Ainsi, les termes d'indexation sont sélectionnés parmi les termes du corpus qui sont en corrélation avec les termes de la requête en plus des termes sélectionnés statiquement comme dans le modèle vectoriel standard.

Par exemple, pour un utilisateur formulant la requête "java", l'ensemble des termes d'indexation sera constitué de termes sélectionnés de manière dynamique (c-à-d les termes qui sont en corrélation avec le mot "java" dans le corpus documen-

taire, exemple : "langage", "programmation", "objet", "applet", "swing",..), et des termes sélectionnés au préalable de manière statique comme dans le modèle vectoriel standard . Ensuite, contrairement au modèle vectoriel standard, on associe au vecteur requête, des poids aux termes sélectionnés dynamiquement.

Plus formellement, $DIIR$ est un modèle de recherche documentaire étendant le modèle vectoriel standard. Il est défini par le 5-uplet $< X, Q, T, s, f >$, où X représente le corpus documentaire, Q est l'ensemble des requêtes, T représente l'ensemble des termes d'indexation, s est la fonction de similarité et f la fonction de construction des termes d'indexation, tel que pour une requête donnée q nous avons $T = f(X, q)$.

L'algorithme ci-dessous décrit le processus de recherche documentaire associé à une nouvelle requête q.

Algorithme 1 : Algorithme de Recherche documentaire
Données :
q : requête utilisateur
X : corpus documentaire
début
1. construction du graphe G de fréquence d'occurrence et de co-occurrence
2. construction de l'ensemble de termes T_{static}
3. **pour** *chaque requête utilisateur* q **faire**
   1. construction de l'ensemble des termes d'indexation $T = T_{static} \bigcup T_{dynamic}$
   2. construction du vecteur de pondération V_{weight}
   3. représentation vectorielle de la requête q et de l'ensemble des documents du corpus sur l'ensemble T des termes d'indexation
   4. enrichissement du vecteur requête q et nouvelle pondération des vecteurs de documents du corpus
   5. calcul de similarité entre le vecteur requête enrichi et l'ensemble des vecteurs de documents
   6. proposition à l'utilisateur des documents les plus similaires à la requête

fin

Dans la suite de cette section, nous décrivons les étapes principales de l'algorithme.

4.2.1 Construction du graphe G de fréquence d'occurrence et de co-occurrence

Afin de pouvoir extraire les termes d'indexation qui sont en corrélation avec la requête, un graphe G de fréquence d'occurrence et de co-occurrence modelisant le corpus X est construit.

Plus formellement $G =< V, E >$ est un graphe étiqueté tel que :

1. $V = \{(t_1, f_1)..(t_n, f_n)\}$ représente l'ensemble des sommets de G, où chaque sommet (t_i, f_i) est représenté par un terme t_i et sa fréquence f_i.
2. $E = \{(t_i, t_j, fco(t_i, t_j))/t_i, t_j \in V\}$ est l'ensemble des arêtes de G, où $fco(t_i, t_j)$ représente la fréquence de co-occurrence entre les termes t_i et t_j.

Deux termes t_i, t_j sont en co-occurrence, s'ils apparaissent en même temps dans le même contexte. Un contexte de co-occurrence peut correspondre à une phrase, un paragraphe, ou même l'ensemble du document [BRC99, MADP04]. Étant donné que nous considérons que les éléments pertinents sont généralement proches dans un document, nous considérons dans notre modèle qu'un contexte correspond à une phrase. La fréquence de co-occurrence correspond au nombre d'occurrence de cette co-occurrence. Ainsi, dans G, $fco(t_i, t_j)$ représente la fréquence de co-occurrence des termes t_i, t_j dans l'ensemble des phrases des documents du corpus.
Au cours de la construction automatique du graphe G, tous les termes du corpus sont pris en considération (sauf les mots vides), le processus d'extraction des termes se fait par :

1. Identification des termes (segmentation lexicale).
2. Élimination des mots vides (déterminants, articles, prépositions, ..).
3. Réduction des termes en leurs racines (Stemmatisation ou lemmatisation).

4.2.2 Construction de l'ensemble T des termes d'indexation

L'ensemble T des termes d'indexation est composé de $T = T_{static} \bigcup T_{dynamic}$.

1. Les termes de l'ensemble T_{static} sont sélectionnés de manière statique et dépendent seulement de l'ensemble X. L'ensemble des termes T_{static} est construit de manière classique et ces termes sont choisis de manière à être les plus discriminants possible. Dans notre contexte, le critère de sélection des termes T_{static} est la fréquence en documents.

2. Les termes de l'ensemble $T_{dynamic}$ sont sélectionnés de manière dynamique et varient en fonction de la requête (Algorithme 2). Ils sont sélectionnés parmi les termes qui sont en corrélation avec les termes de la requête. Ainsi pour chaque nouvelle requête q, un nouvel ensemble $T_{dynamic}$ est construit.

Algorithme 2 : construction de l'ensemble $T_{dynamic}$
Données :
q : requête utilisateur
G : graphe de fréquence d'occurrence et de co-occurrence du corpus documentaire,
β : constante représentant le seuil de sélection des termes, avec $0 \leq \beta \leq 1$
l'ensemble $T_{dynamic}$ de termes d'indexation
début
1. $T_{dynamic} \leftarrow \emptyset$;
2. **pour** *chaque terme* t_i *de* q **faire**
   pour *chaque terme* t_j *de* G *tel que* $fco(t_i, t_j) > 0$ **faire**
   si $\frac{(fco(t_i,t_j))^2}{f_{t_i} \times f_{t_j}} > \beta$ **alors**
   $T_{dynamic} = T_{dynamic} \cup \{t_j\}$

fin

4.2.3 Construction du vecteur de pondération $\vec{V}_{weight}$

Afin d'enrichir le vecteur requête initial et de pondérer les vecteurs documents du corpus X, un vecteur de pondération $\overrightarrow{V}_{weight} = (w_1..w_{|T|})$ est calculé, où à chaque terme t_i de T un poids w_i est associé.

Dans $\overrightarrow{V}_{weight}$ un poids non nul est affecté aux termes qui sont en corrélation avec les termes de la requête (c-à-d $T_{dynamic}$). Comme le montre l'algorithme 3, le poids w_i du terme t_i dépend de 3 facteurs :
1. La corrélation de t_i avec les termes de la requête (i.e. $\frac{(fco(t_i,t_j))^2}{f_{t_i} \times f_{t_j}}$, où t_j est un terme de la requête).
2. La fréquence en documents des termes de la requête auxquels t_i est en corrélation.
3. Le nombre de termes de la requête avec lesquels t_i est en corrélation.

Algorithme 3 : Construction du vecteur de pondération $\vec{V}_{weight} = (w_1..w_{|T|})$
Données : T : ensemble des termes d'indexation,
G : graphe de fréquence d'occurrence et de co-occurrence du corpus documentaire
β : constante représentant le seuil de sélection des termes, avec $0 \leq \beta \leq 1$,
$\vec{V}_{weight} = (w_1..w_{|T|})$: vecteur de pondération
début
1. $\overrightarrow{V}_{weight} = \{w_i = 0 / i = 1..|T|\}$
2. $rep = \{rep_i = 0 / i = 1..|T|\}$
3. **pour** *chaque terme* t_j *de* q **faire**
   pour *chaque terme* t_i *de* $T_{dynamic}$ *telque* $\frac{(fco(t_i,t_j))^2}{f_{t_i} \times f_{t_j}} > \beta$ **faire**
   $w_i = w_i + df(t_j) \times \frac{(fco(t_i,t_j))^2}{f_{t_i} \times f_{t_j}}$
   $rep_i = rep_i + 1$;
4. **pour** $(i = 1..|T|)$ **faire**
   $w_i = w_i \times \exp(rep_i)$
fin

$df(t_j)$ est la fréquence en documents du terme t_j.
rep_i est le nombre de termes de la requête avec lesquels t_i est en corrélation.

4.2.4 Représentation vectorielle de la requête

La requête q est initialement représentée par un vecteur $\overrightarrow{q}$ indexé sur l'ensemble des termes T et pondéré par *TF-IDF*. Ensuite $\overrightarrow{q}$ est enrichi par le vecteur $\overrightarrow{V}_{weight}$. Ainsi nous obtenons un nouveau vecteur $\overrightarrow{q'}$ pour lequel des poids non nuls sont associés aux termes qui sont en corrélation avec les termes de la requête initiale q (c-à-d $T_{dynamic}$).

$$\overrightarrow{q'} = \alpha \times \frac{\overrightarrow{q}}{|\overrightarrow{q}|} + (1-\alpha) \times \frac{\overrightarrow{V}_{weight}}{|\overrightarrow{V}_{weight}|}$$

$\overrightarrow{q}$: vecteur requête initiale,
$|\overrightarrow{q}|$: norme euclidienne du vecteur $\overrightarrow{q}$,
$\overrightarrow{V}_{weight}$: vecteur pondération,
$|\overrightarrow{V}_{weight}|$: norme euclidienne du vecteur de pondération $\overrightarrow{V}_{weight}$,
α : constante comprise entre $0 \leq \alpha \leq 1$, permettant l'hybridation entre la requête initiale normalisée $\frac{\overrightarrow{q}}{|\overrightarrow{q}|}$ et le vecteur de pondération normalisé $\frac{\overrightarrow{V}_{weight}}{|\overrightarrow{V}_{weight}|}$.

4.2.5 Représentation vectorielle des documents

Chaque document d est initialement représenté par un vecteur $\overrightarrow{d}$ indexé sur l'ensemble des termes T et pondéré par *TF-IDF*. Ensuite un nouveau vecteur $\overrightarrow{d'}$ qui est, la pondération de $\overrightarrow{d}$ par le vecteur $\overrightarrow{V}_{weight}$, est calculé.

Dans $\overrightarrow{d'}$ les poids des termes de $T_{dynamic}$ sont ajustés en fonction de leurs importances (c-à-d leurs corrélations avec les termes de la requête) et les autres termes sont mis à 0.

$$\overrightarrow{d'} = (w_i * w'_i) \text{ avec } i = 1 \text{ à } \parallel T \parallel, \text{ tel que } w_i \in \overrightarrow{d} \text{ et } w'_i \in \overrightarrow{V}_{weight}$$

Ensuite, un nouveau vecteur document $\overrightarrow{d''}$ est calculé comme suit :

$$\overrightarrow{d''} = \alpha \times \frac{\overrightarrow{d}}{|\overrightarrow{d}|} + (1-\alpha) \times \frac{\overrightarrow{d'}}{|\overrightarrow{d'}|}$$

L'objectif de $\overrightarrow{d''}$ est de prendre aussi en considération le poids des termes initiaux de la requête et des termes n'appartenant pas à $T_{dynamic}$.

$|\overrightarrow{d}|$: norme euclidienne du vecteur $\overrightarrow{d}$,
$|\overrightarrow{d'}|$: norme euclidienne du vecteur document pondéré $\overrightarrow{d'}$,
α : constante comprise entre $0 \leq \alpha \leq 1$, permettant l'hybridation entre le vecteur document initial normalisé $\frac{\overrightarrow{d}}{|\overrightarrow{d}|}$ et le vecteur pondéré $\frac{\overrightarrow{d'}}{|\overrightarrow{d'}|}$.

La recherche documentaire s'effectue par un calcul de similarité entre le vecteur requête enrichi $\overrightarrow{q'}$ et chaque vecteur document $\overrightarrow{d''}$. Ainsi, les documents les plus similaires à la requête sont proposés à l'utilisateur.

4.2.6 Illustration

Soit un utilisateur défini par son profil $P =< id, G >$. Considérons que cet utilisateur recherche des documents sur la France via la requête "France". La figure 3 illustre une partie du graphe G du profil utilisateur P.
Les termes d'indexation sont choisis parmi les voisins du terme "France" dans le graphe G modélisant le profil utilisateur. Ainsi, avec par exemple un seuil $\beta = 0.01$, en appliquant l'algorithme de construction des termes d'indexation (algorithme 3), nous obtenons :

$\frac{fco("france","football")^2}{f_{"france"} \times f_{"football"}} = \frac{10^2}{30 \times 30} = 0.111 > \beta \Rightarrow$ sera sélectionné
$\frac{fco("france","zidane")^2}{f_{"france"} \times f_{"zidane"}} = \frac{15^2}{30 \times 40} = 0.187 > \beta \Rightarrow$ sera sélectionné
$\frac{fco("france","europe")^2}{f_{"france"} \times f_{"europe"}} = \frac{3^2}{30 \times 10} = 0.030 > \beta \Rightarrow$ sera sélectionné
$\frac{fco("france","paris")^2}{f_{"france"} \times f_{"paris"}} = \frac{10^2}{30 \times 20} = 0.166 > \beta \Rightarrow$ sera sélectionné
$\frac{fco("france","cuisine")^2}{f_{"france"} \times f_{"cuisine"}} = \frac{5^2}{30 \times 15} = 0.055 > \beta \Rightarrow$ sera sélectionné
$\frac{fco("france","java")^2}{f_{"france"} \times f_{"java"}} = \frac{1^2}{30 \times 13} = 0.002 < \beta \Rightarrow$ sera non sélectionné
$\frac{fco("france","avoir")^2}{f_{"france"} \times f_{"avoir"}} = \frac{10^2}{30 \times 30} = 0.006 < \beta \Rightarrow$ sera non sélectionné

L'ensemble des termes d'indexation choisi est :

$$T = \{"France","europe","cuisine","paris","football","zidane"\}$$

Le vecteur requête indéxé par rapport à l'ensemble des termes T est donc :

$$q = (1, 0, 0, 0, 0, 0)$$

La matrice profil-requête sera alors :

$$M_T = \begin{bmatrix} 0 & 3 & 5 & 10 & 10 & 15 \\ 3 & 0 & 0 & 0 & 5 & 7 \\ 5 & 0 & 0 & 5 & 0 & 0 \\ 10 & 0 & 5 & 0 & 0 & 0 \\ 10 & 5 & 0 & 0 & 0 & 10 \\ 15 & 7 & 0 & 0 & 10 & 0 \end{bmatrix}$$

Par exemple, l'élément $M_T[1][3]$ correspond à la fréquence de co-occurrences entre les termes "france" et "cuisine".

Ainsi la requête transformée devient :

$$q' = (1-\alpha)\times\frac{q}{|q|}+\alpha\times\frac{q\times M_T}{|q\times M_T|} = (1-\alpha)\times(1,0,0,0,0,0)+\frac{\alpha}{\sqrt{459}}\times(0,3,5,10,10,15)$$

Avec $\alpha = 0.5$ nous obtenons : $q' = (0.5, \frac{1.5}{\sqrt{459}}, \frac{2.5}{\sqrt{459}}, \frac{5}{\sqrt{459}}, \frac{5}{\sqrt{459}}, \frac{7.5}{\sqrt{459}})$

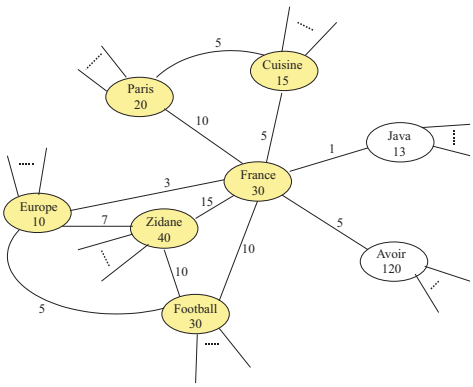

FIG. 4.1 – partie du graphe modélisant le profil utilisateur

La proposition de document à l'utilisateur se fera par le calcul de similarité entre le vecteur q' et l'ensemble des documents du corpus indexés sur l'ensemble des termes T. Ainsi les documents les plus similaires à q' seront proposés à l'utilisateur.

4.3 Experimentation

4.4 Application de l'approche pour le Web : l'Agent PAWebSearch

4.5 PAWebSearch Architecture

Pour répondre de la manière la plus adaptée aux besoins de l'utilisateur, il est indispensable dans un premier temps d'effectuer une analyse des différents documents

parcourus par l'utilisateur afin d'en extraire son profil. A partir de ce profil, la personnalisation consiste donc à vérifier si les documents retournés par les moteurs sont adaptés ou non. Dans notre cadre, nous réalisons la recherche d'information sur le Web au travers de l'agent autonome *PAWebSearch*. Dans cette section, nous présentons l'architecture fonctionnelle de celui-ci.

L'agent *PAWebSearch* est chargé d'effectuer les deux tâches suivantes :

1. *Apprentissage du profil utilisateur* : en fonction des différentes actions menées par l'utilisateur et en fonction de chaque document consulté (page Web), *PAWebSearch* met à jour automatiquement son profil.
2. *Personnalisation des résultats de la recherche d'information* : A chaque requête de recherche documentaire via un moteur de recherche (*Google, Yahoo* .. etc), *PAWebSearh* récupère la requête utilisateur ainsi que les résultats retournés par le moteur et classe ces résultats en fonction du profil de l'utilisateur.

Comme nous pouvons le constater (C.f. Figure 1), *PAWebSearch* est constitué de trois modules principaux :

1. *Un Proxy* : Il s'agit de l'interface privilégiée entre le navigateur, le Web, et les sous systèmes d'apprentissage et de filtrage. Toutes les requêtes de l'utilisateur sont envoyées au Proxy qui est chargé d'exécuter la requête sur le Web et de mettre à jour le profil utilisateur. Les réponses à des requêtes de recherche d'information, i.e. à des moteurs de recherche, sont envoyées au système de filtrage qui est chargé d'adapter les résultats en fonction du profil utilisateur.
2. *Un module d'apprentissage du profil utilisateur* : il modélise l'utilisateur via ses actions, en effet ce module surveille les actions de l'utilisateur et à chaque nouveau document consulté met à jour le profil.
3. *Un module de filtrage* : il filtre les résultats de requêtes de recherche d'information en fonction du profil utilisateur.

Le principe général est le suivant. A partir d'une requête q effectuée par l'utilisateur sur un moteur de recherche, nous récupérons tous les résultats au travers du *Proxy*. Une première analyse des connaissances de l'utilisateur (i.e. du profil) permet d'obtenir un ensemble de termes d'indexation T, constitué de termes de la requête q, enrichie par les termes associés régulièrement aux termes de la requête q. Ces associations sont déterminées au préalable par l'analyse des co-occurrences de termes dans les documents consultés par l'utilisateur.

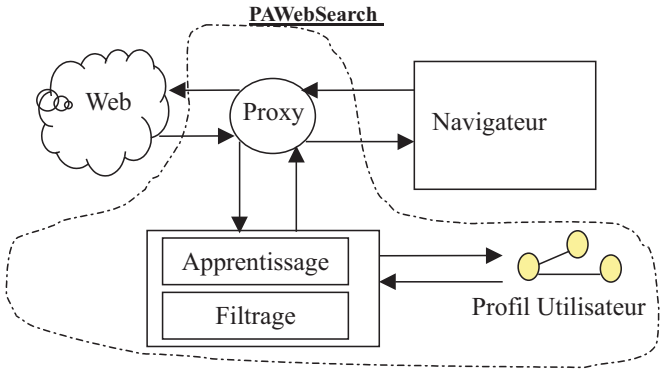

FIG. 4.2 – Architecture fonctionnelle générale de l'agent PAWebSearch

Dans notre contexte, les requêtes et les documents que nous manipulons sont représentés par des vecteurs dans le cadre du modèle vectoriel standard [SG83]. Ainsi, nous indexons, dans un premier temps, la requête q ainsi que tous les documents retournés sur l'ensemble des termes d'indexation T. Dans un second temps un nouveau vecteur requête q' enrichi par le profil utilisateur est calculé. La proposition de documents à l'utilisateur se fait par le calcul de similarités entre les documents retournés par le moteur et le nouveau vecteur requête q', ainsi les documents les plus similaire sont retournés à l'utilisateur.

La Figure 4 représente les résultats de la requête "java" via *Google*. Les résultats de la requête "java" en utilisant *PAWebSearch* sont décrits sur la Figure 5, cette requête est effectuée après un apprentissage du profil de l'utilisateur sur des documents concernant principalement *l'île java*. Il faut remarquer que les trois premières *urls* proposées à l'utilisateur par *PAWebSearch* sont des documents parlant de *l'île java*. Alors que dans les résultats de *Google*, le premier document parlant de *l'île java* n'apparaît qu'à la cinquième position et le second qu'à la 47^{eme} position.

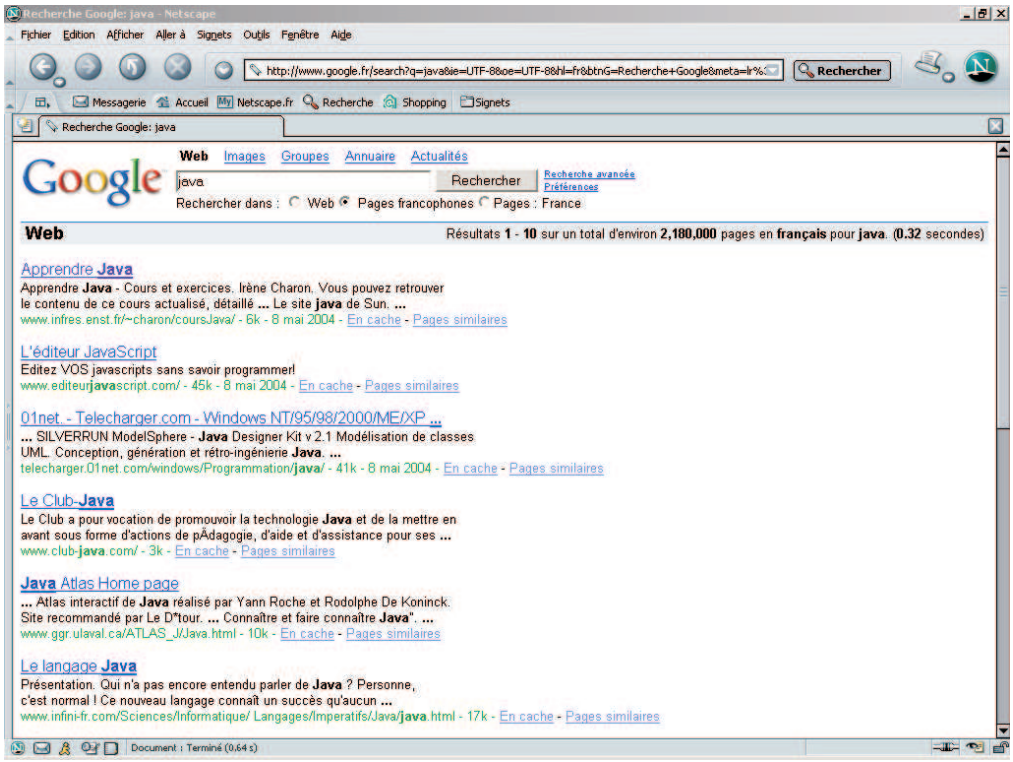

FIG. 4.3 – Résultats de *Google* pour la requête "java"

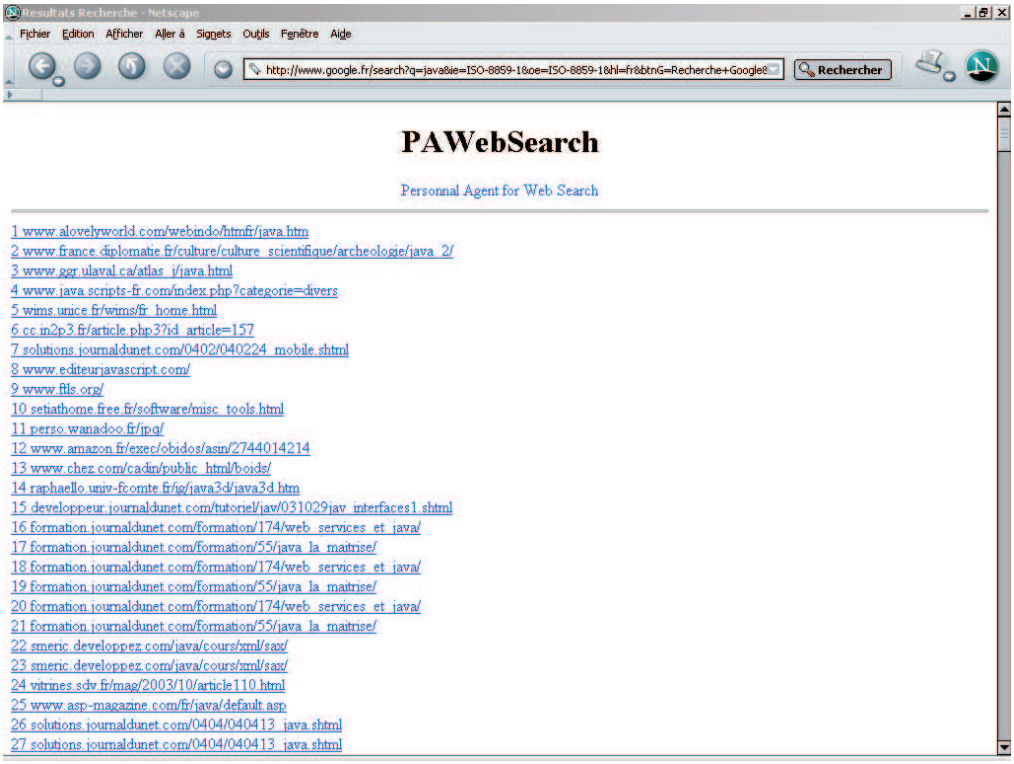

FIG. 4.4 – Résultats de *PAWebSearch* pour la requête "java"

Chapitre 5

Modélisation dynamique et temporelle de l'utilisateur pour la recommandation de documents textuels

Sommaire

5.1 Introduction

Avec le développement d'Internet et les nouveaux moyens de stockage de données, les serveurs de documents en ligne regorgent d'énormes quantités de documents de différentes thématiques. La recherche en ligne, par l'utilisateur, de documents répondants à son intérêt est un travail fastidieux. Les moteurs de recherche peuvent être d'une grande utilité pour la recherche de documents pertinents mais ils nécessitent de formuler de nouvelles requêtes à chaque fois que l'utilisateur a besoin de nouveaux documents. Récemment, des efforts considérables ont été déployés pour résoudre ce problème et cela par le développement de systèmes capables de proposer des documents adaptés à l'utilisateur, sans que ce dernier ne formule de requêtes. Ces systèmes prennent en compte le profil de l'utilisateur pour proposer des documents personnalisés. Cependant, même si des systèmes de filtrage documentaires basés sur le profil utilisateur existent, ils ne prennent pas en compte l'évolution dans le temps des classes de documents consultés par l'utilisateur. Dans ce papier nous proposons une nouvelle approche d'apprentissage du profil long terme de l'utilisateur pour le filtrage de documents textuels. Cette approche est basée sur l'analyse de l'évolution dans le temps des classes de documents consultés par l'utilisateur. Dans ce cadre, les documents consultés sont classés de manière dynamique et nous analysons ensuite la répartition, dans le temps, de ces classes de documents. Le but de notre approche est de déterminer le mieux possible les classes d'intérêts de l'utilisateur, cela en donnant plus d'importance aux classes de documents régulièrement consultées qu'à celles concentrées sur une courte période. Notre approche ne requiert donc pas que l'utilisateur fournisse de manière explicite des informations au système.

Nous avons développé *LUCI* un système permettant la personnalisation de documents en ligne via notre approche. *LUCI* apprend le profil de l'utilisateur via les documents consultés par celui-ci et lui propose des documents de manière dynamique, i.e à chaque fois que l'utilisateur modifie son profil (consulte un nouveau document) le système lui propose une collection de documents adaptés à son profil.

L'article est organisé de la manière suivante. Dans la section 2 nous présentons les problématiques de la modélisation des intérêts long terme de l'utilisateur pour le filtrage personnalisé de documents textuels. La section 3 présente l'architecture fonctionnelle du système *LUCI*. Ensuite dans la section 4 nous décrivons de manière détaillée notre approche pour la modélisation de l'intérêt long terme de l'utilisateur basée sur la répartition dans le temps des classes de documents consultées. La section 5 présente le processus de filtrage documentaire basé sur notre modèle utilisateur.

Nous présentons à la section 6 une série d'expériences sur un corpus documentaire de référence, nous montrons que notre approche permet de détecter et de proposer des documents d'intérêt régulier. Un bref état de l'art sur la modélisation de l'utilisateur est proposé à la section 7. Enfin, dans la section 8, nous concluons en résumant les avantages et présentons les perspectives associées au modèle.

5.2 Problèmatique

L'objectif de notre proposition est l'apprentissage du profil long terme de l'utilisateur pour une personnalisation efficace de documents textuels. Pour atteindre cet objectif, nous classons les documents consultés par l'utilisateur et nous analysons l'évolution de ces classes dans le temps. L'idée générale est de considérer qu'une classe de documents consultée régulièrement par l'utilisateur a plus d'intérêt à long terme pour ce dernier qu'une classe de documents concentrée sur une courte période. Par exemple, un utilisateur ayant des actions en bourse consulte tous les jours un ou deux documents sur la thématique *finance*. Ce même utilisateur, ayant voulu faire un exposé sur les dinosaures, a consulté plus de 100 documents sur une période de deux jours sur la thématique *dinosaure*. On considère, qu'à long terme (un mois par exemple) la thématique *finance* a plus d'intérêt pour l'utilisateur que la thématique *dinosaure*, alors que le nombre de documents consultés de la thématique *dinosaure* est bien plus important que le nombre de documents consultés de la thématique *finance*. La thématique *finance* représente un intérêt régulier pour l'utilisateur alors que la thématique *dinosaure* est d'un intérêt spontané, i.e concentrée sur une très courte période relativement à la thématique *finance*. A long terme (un mois dans cet exemple) nous devons être capable de proposer à l'utilisateur des documents de la thématique *finance* car elle est d'un intérêt régulier, et de ne pas proposer des documents de la thématique *dinosaure*, car elle ne représente qu'un intérêt spontané.

Dans notre contexte, l'utilisateur est connecté à une base documentaire en ligne, cette dernière contient une collection de documents de différentes thématiques. Le corpus documentaire est indexé et à chaque document est associé un vecteur. L'utilisateur est modélisé à partir de l'ensemble des documents (représentés dans notre modèle par des vecteurs), consultés au cours du temps. La problématique de la personnalisation de documents à long terme consiste à trouver un modèle capable de prendre en compte l'évolution dans le temps des classes de documents consultés par l'utilisateur et de détecter au mieux les classes d'intérêts réguliers pour une meilleur

personnalisation de documents.

5.3 Architecture du système LUCI

Le but du système *LUCI* est d'apprendre de manière dynamique les intérêts "long terme" de l'utilisateur et de proposer à ce dernier des documents adaptés à ses intérêts. *LUCI* modélise l'utilisateur via les documents consultés, et à partir du modèle lui propose une collection de documents. A chaque nouveau document consulté, *LUCI* met à jour le modèle utilisateur et lui propose une nouvelle collection de documents. Comme le montre la figure 1, *LUCI* est composé de deux sous-systèmes et d'une structure définissant le modèle utilisateur :

1. *Sous-système de modélisation de l'utilisateur* : il modélise l'utilisateur via ses actions. En effet, ce sous-système surveille les actions de l'utilisateur et à chaque document consulté par ce dernier, il met à jour le modèle utilisateur.
2. *Sous-système de filtrage documentaire* : il propose des documents en fonction du modèle utilisateur. A chaque mise à jour du modèle utilisateur (nouveau document consulté), ce sous-système propose une nouvelle collection de documents.
3. *Le modèle utilisateur* : il s'agit d'une structure complexe modélisant les intérêts long terme de l'utilisateur.

Il existe de nombreux modèles vectoriels pour représenter les documents [Cha90, **?**]. Dans notre contexte, nous considérons que les documents sont représentés par le modèle vectoriel classique [Sal71, SG83].
L'extraction des termes (mots-clés) se fait par :

- La reconnaissance des termes (segmentation lexicale),
- L'élimination des termes peu intéressants, une fois isolés de leur contexte (suivant une liste prédéfinie),
- La réduction des termes en leur racine[1],
- Le choix des termes les plus fréquents (10% des mots clés existant).

Pour réduire l'avantage des documents long sur les documents courts, les vecteurs représentant les documents sont normalisés [**?**, Sin97]. Soit D un document textuel représenté par l'ensemble des tuples $\{(t_1, w_1), ..(t_n, w_n)\}$ où t_i est un terme

[1]forme tronquée (stem) ou lemme.

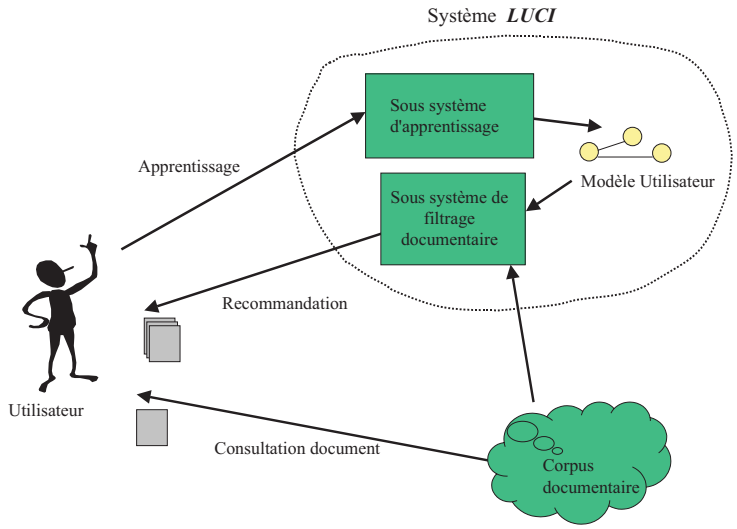

FIG. 5.1 – Architecture fonctionnelle générale du système *LUCI*

(mot) appartenant à D et w_i est le poids de t_i. Les vecteurs mots clés utilisés sont normalisés de la manière suivante :

$$w_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$

où f_i représente la fréquence du terme t_i.

Nous utilisons, pour mesurer la similarité, la mesure du *cosinus* [SG83]. Étant donné deux documents définis respectivement par les vecteurs D_i et D_j, la similarité du *cosinus* entre ces deux documents est donnée par la formule :

$$SIM(D_i, D_j) = COS\ \theta(D_i, D_j) = (D_i * D_j)/(\mid D_i \mid^{1/2} * \mid D_j \mid^{1/2})$$

5.4 Modélisation de l'intérêt long-terme

Dans cette section nous décrivons notre approche pour la modélisation de l'intérêt long-terme. Notre motivation derrière cette modélisation est de capturer le mieux possible l'intérêt général de l'utilisateur. Ainsi, nous considérons qu'une classe régulièrement consultée (intérêt permanent) a plus d'intérêt qu'une classe concentrée sur une courte période (intérêt spontané).

notre approche est basée sur :

1. Une classification dynamique des documents consultés par l'utilisateur ;
2. L'association à chaque classe d'un vecteur, ce dernier correspond à la somme des vecteurs de documents de la classe ;
3. Un poids associé à chaque classe, celui-ci détermine la répartition dans le temps des documents de la classe ;
4. Un vecteur nommé LTV (Long Term Vector) déterminant l'intérêt long terme de l'utilisateur est calculé à partir des vecteurs de classes, et de leurs répartitions ;
5. Un calcul de similarité entre les documents du corpus et le vecteur LTV. Les documents les plus similaires sont proposés à l'utilisateur.

Dans la suite de cette section. Nous commençons par décrire de manière détaillée la structure du modèle long terme. Ensuite nous décrivons la méthode de classification dynamique de documents, suivi de la méthode de calcul de la répartition des documents d'une classe. Enfin, nous présentons le calcul du vecteur LTV ainsi qu'un exemple illustrant notre modèle.

5.4.1 Structure du modèle long terme

Le modèle utilisateur est défini par le tuple $X =< id, S >$ où :
id identifie de manière unique l'utilisateur et S est l'ensemble des classes de documents consultés par l'utilisateur id, i.e $S = \{C_1, ...C_n\}$ où n est le nombre de classes.

Chaque classe C_i, i=1 à n, est définie par le tuple $C_i = \{V, Rep_{C_i}, V_{C_i}\}$, tel que :

1. V : est un ensemble de tuples contenant les documents consultés de la classe C_i (vecteurs) et leurs positions (dans le temps).

$$\{(V_1^{C_i}, Pos_1)..(V_{\|C_i\|}^{C_i}, Pos_{\|C_i\|})\}$$

$\|C_i\|$ est le nombre de documents de la classe C_i,
$V_j^{C_i}$: le J^{eme} document de la classe C_i,
La position d'un document Pos_i représente son ordre d'apparition dans le temps par rapport aux documents consultés.

2. Rep_{C_i} : répartition dans le temps des documents de la classe C_i (voir section 4.3)
3. V_{C_i} : le vecteur de la classe C_i, avec :

$$V_{C_i} = \sum_{j=1}^{\|C_i\|} V_j^{C_i}$$

Nous allons maintenant examiner comment les classes sont déterminées et comment les documents sont associés à ces classes.

5.4.2 Classification des documents consultés par l'utilisateur

L'algorithme ci-dessous décrit le processus de classification des documents consultés par l'utilisateur. La création des classes se fait de manière dynamique. Initialement, le nombre de classes est nul ($S = \emptyset$). Au premier document consulté par l'utilisateur, l'algorithme de classification crée la classe C_1. Ensuite, pour chaque document consulté, l'algorithme calcule la similarité entre le document et l'ensemble des classes existantes. Si le document est proche de l'une des classes existante alors le document est ajouté à celle ci. Sinon l'algorithme crée une nouvelle classe et associe ce document à cette dernière.

L'algorithme ci-dessous reçoit en entrée un nouveau document, représenté par un vecteur V_D et le modèle utilisateur $X =< id, S >$. En sortie, il associe le document V_D et sa position à une classe.

$\|S\|$: représente le nombre de classes contenu dans le modèle utilisateur,
SIM : est la fonction de similarité entre deux vecteurs,
α : constante représentant un seuil de similarité.

5.4.3 Calcul de la répartition des classes de documents

Pour le calcul de la répartition dans le temps des classes de documents consultés par l'utilisateur, on associe à chaque classe de documents consultés par l'utilisateur un nuage de points représenté dans un espace à deux dimensions. Chaque point

Algorithme 1 : Algorithme de classification dynamique
Input : Document V_D
Pos_{V_D} : position du document V_D
modèle utilisateur $X =< id, S >$,
Output : association du document V_D à une classe de documents
begin
  if $\|S\| = 0$ **then**
    création de la classe C_1
    Ajouter (V_D, Pos_{V_D}) à la classe C_1
  else
    $k \leftarrow 1$
    for $j = 1 \quad à \quad j = \|S\|$ **do**
      if $SIM(V_{C_j}, V_D) > SIM(V_{C_k}, V_D)$ **then**
        $k \leftarrow j$
    if $SIM(V_{C_k}, V_D) > \alpha$ **then**
      Ajouter (V_D, Pos_{V_D}) à la classe C_k
      else
        créer une nouvelle classe $C_{\|S\|+1}$
        Ajouter (V_D, Pos_{V_D}) à la classe $C_{\|S\|+1}$
end

représente un document de la classe. A chaque document on associe deux coordonnées :

1. position du document dans l'ensemble des documents, c'est à dire l'ordre de consultation de ce document par l'utilisateur,
2. position du document par rapport aux documents de sa classe, c'est à dire l'ordre d'ajout de ce document à sa classe.

Une droite de regression (Δ) des moindres carrés du nuage de points est calculée. La répartition d'une classe de documents est donnée par la formule :

$$Rep_{C_i} = \frac{Pos_{\|C_i\|} - Pos_1 + 1}{N} * \frac{1}{1 + \sum_{j=1}^{j=\|C_i\|} Distance(D_j, \Delta)}$$

Rep_{C_i} : répartition de la classe C_i,
$Pos_{\|C_i\|}$: position du dernier document consulté appartenant à la classe C_i,
Pos_1 : position du premier document consulté, appartenant à la classe C_i ,

Δ : droite de régression des moindres carrés du nuage de points de la classe,
D_j : coordonnées du j ème document de la classe C_i,
N : nombre de documents consultés par l'utilisateur,
$Distance(D_j, \Delta)$: distance entre le point D_j et la droite Δ.

5.4.4 Calcul du vecteur long terme LTV

A partir du modèle long terme, un vecteur définissant l'intérêt long terme est calculé comme suit :

$$LTV = \sum_{1}^{\|S\|} Rep_{Ci} * V_{Ci}$$

où $\|S\|$ est le nombre de classes

La proposition de documents relatif à l'intérêt long terme de l'utilisateur se fait par le calcul de similarités entre les documents du corpus et le vecteur LTV. Si la similarité entre LTV et un document est supérieur à un certain seuil alors le document est proposé à l'utilisateur.

5.4.5 Exemple

Soit un utilisateur ayant consulté une suite de documents : $D_1, D_2, .., D_9$, au 9ème document consulté, l'algorithme de classification dynamique ayant formé 3 classes :
$C_1 = \{D_1, D_5, D_9\}$
$C_2 = \{D_2, D_3, D_4\}$
$C_3 = \{D_6, D_7, D_8\}$
Le modèle utilisateur sera défini par $X =< id, S >$ tel que :
$S = \{C_1, C_2, C_3\}$ avec :

1. $C_1 = \{V, Rep_{C_1}, V_{C_1}\}$, $V = \{(D_1, 1), (D_5, 5), (D_9, 9)\}$
   $Rep_{C_1} = \frac{9-1+1}{9} \star \frac{1}{1+0} = 1$
   $V_{C_1} = D_1 + D_5 + D_9$
2. $C_2 = \{V, Rep_{C_2}, V_{C_2}\}$, $V = \{(D_2, 2), (D_3, 3), (D_4, 4)\}$
   $Rep_{C_2} = \frac{4-2+1}{9} \star \frac{1}{1+0} = \frac{3}{9}$
   $V_{C_2} = D_2 + D_3 + D_4$
3. $C_3 = \{V, Rep_{C_3}, V_{C_3}\}$, $V = \{(D_6, 6), (D_7, 7), (D_8, 8)\}$
   $Rep_{C_3} = \frac{8-6+1}{9} \star \frac{1}{1+0} = \frac{3}{9}$
   $V_{C_3} = D_6 + D_7 + D_8$

$$LTV = (1 \star V_{C_1}) + (\frac{3}{9} \star V_{C_2}) + (\frac{3}{9} \star V_{C_3})$$

On remarque que le nombre de documents des classes C_1, C_2, C_3 est identique, cependant dans le calcul du vecteur LTV la classe C_1 est avantagée, car elle est d'un intérêt régulier contrairement aux classes C_2 et C_3 qui sont d'un intérêt spontané.

5.5 Filtrage de documents textuels

La proposition de documents à l'utilisateur se fait par le calcul de similarités entre les documents du corpus documentaire et le vecteur LTV. A chaque fois que l'utilisateur consulte un document, son profil est modifié, ainsi le système lui propose une nouvelle collection de documents. L'algorithme ci-dessous décrit le processus de filtrage documentaire associé à la consultation d'un nouveau document par l'utilisateur.

Algorithme 2 : Algorithme de filtrage documentaire
Input :
Document V_D
Pos_{V_D} : position du document V_D,
modèle utilisateur $X =< id, S >$,
α : constante représentant le seuil de similarité
Output : proposition d'un ensemble de documents à l'utilisateur
begin
1. Associer le document V_D à une classe de documents (voir Algorithme 1) ;
2. Calculer le vecteur de la classe où se trouve V_D ;
3. Calculer la répartition de l'ensemble des classes ;
4. Calculer le vecteur LTV ;
5. Calculer la similarité entre le vecteur LTV et l'ensemble des documents du corpus documentaire ;
6. Proposer à l'utilisateur les documents dont la similarité est supérieur au seuil α ;

end

5.6 Expérimentation

Une évaluation a été faite pour mesurer la capacité d'apprentissage du système *LUCI*. L'objectif principal est de mesurer la capacité du système à proposer des documents d'intérêts réguliers.

5.6.1 Méthode

Les documents utilisés pour notre étude sont des articles de presse, collectés de 5 journaux en ligne différents sur des périodes différentes. Notre corpus documentaire contient des documents de 6 thématiques différentes de 200 documents chacune. Les thématiques choisies sont : *Économie, Football, Crise-Irakienne, Politique, Cinéma, Informatique.*

Initialement le profil de l'utilisateur est vide. Ensuite celui-ci consulte des documents les uns après les autres. A chaque document consulté, le système *LUCI* lui propose vingt documents (les plus pertinents). A chaque proposition de documents par le système *LUCI* nous faisons une évaluations de ces derniers.

Pour montrer la capacité du système *LUCI* à proposer des documents d'intérêts réguliers, considérons un utilisateur consultant en premier un ensemble de documents de la thématique *informatique* (12 documents), ensuite un ensemble de documents

de la thématique *économie* (30 documents), suivi d'un ensemble de documents de la thématique *football* (24 documents). Ainsi qu'une consultation régulière de documents de la thématique *cinéma* (11 documents). Cette dernière représente un intérêt régulier pour l'utilisateur. Notre motivation est de voir la capacité du système à proposer des documents de la classe *cinéma*.

Nous avons choisi de comparer *LUCI* à l'algorithme *LUCI-NEG*, une implementation de *LUCI* sans prise en compte de la répartition dans le temps des classes de documents.

5.6.2 Analyse des résultats

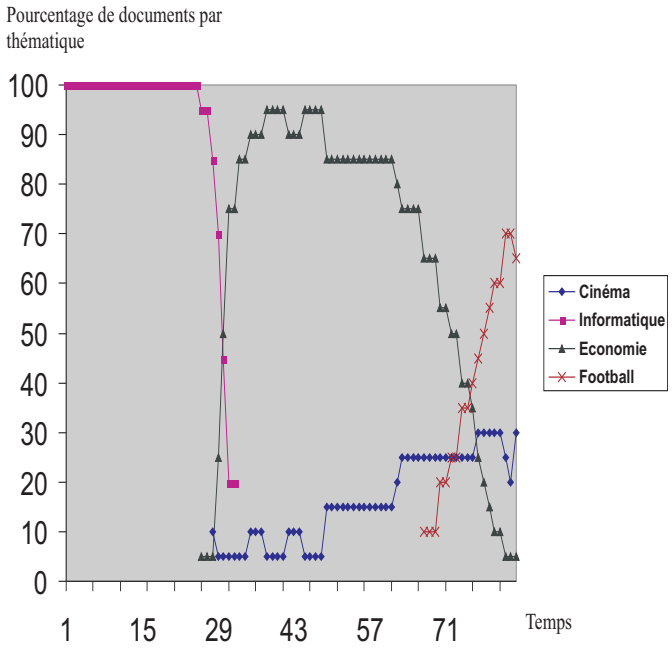

FIG. 5.2 – *LUCI* : Pourcentage de documents par thématique proposés à l'utilisateur

Les figures 2 et 3 montrent les pourcentages de documents par thématique proposés à l'utilisateur, respectivement par le système *LUCI* et l'algorithme *LUIC-NEG*. On remarque qu'à partir de l'itération 29 le système *LUCI* propose des documents de la thématique *Cinéma* (thématique régulièrement consultée), alors que *LUCI-NEG* n'en propose pas. Cela est dû à la capacité du système *LUCI* de détecter les classes de documents d'intérêts réguliers.

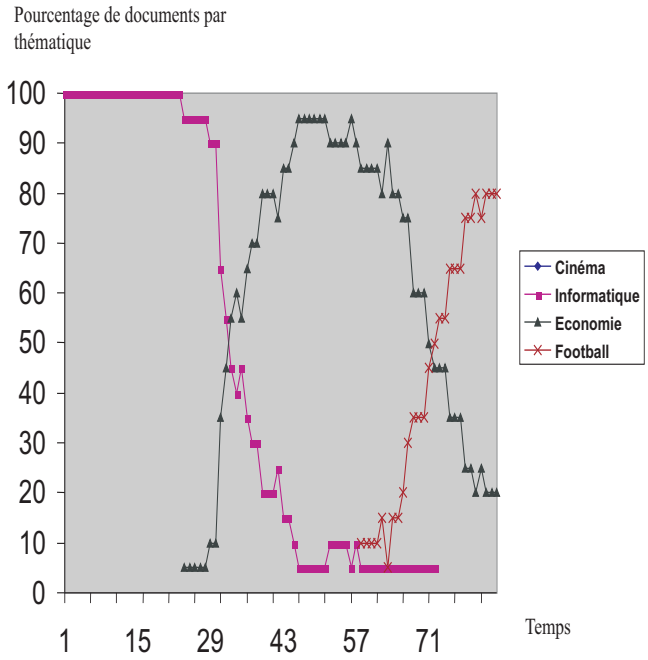

FIG. 5.3 – *LUCI-NEG* : Pourcentage de documents par thématique proposés à l'utilisateur

La thématique *Informatique* est d'un intérêt spontané pour l'utilisateur, en effet l'utilisateur ne s'intéresse plus à cette thématique à partir de l'itération 14. On remarque que *LUCI-NEG* continue à proposer d'une manière importante des documents de cette thématique jusqu'à l'itération 72, alors que le nombre de documents de la thématique *Informatique* proposé par *LUCI* décroît très rapidement pour s'annuler à l'itération 28. Les mêmes constatations sont faites pour la thématique *Économie*. Cela est dû à la capacité de *LUCI* de détecter les classes de documents d'intérêts spontanés et d'en diminuer constamment et rapidement le nombre de documents proposés appartenant à ces classes.

Chapitre 6

Conclusion

Bibliographie

[AF77] R. Attar and A. S. Fraenkel. Local feedback in full-text retrieval systems. *J. ACM*, 24(3) :397–417, 1977.

[All96] J. Allan. Incremental Relevance Feedback for Information Filtering. In *Proceedings of the 19th International ACM-SIGDIR Conference on Research and Developement in Information Retreival*, pages 270–278, 1996.

[AMD+04] R. Arezki, A. Mokrane, G. Dray, P. Poncelet, and D.W. Pearson. LUCI : A Personnalization Documentary System Based on the Analysis of the History of the User's Actions. In *Proceedings of the 6th International Conference On Flexible Query Answering Systems (FQAS 2004)*. LNAI, Springer Verlag, 2004.

[Bal97a] M. Balbanovic. An Adaptative Web Page Recommendation Service. In *Proceeding of the First International Conference on Autonomous Agents*, pages 378–385, 1997.

[Bal97b] M. Balbanovic. An interface for learning multi-topic user profiles from implicit feedback. In *Proceeding of the AAAI-98 Workshop on Recommender Systems*, pages 6–10, 1997.

[BC92] N.J. Belkin and W.B. Croft. Information filtering and information retrieval : two sides of the same coin ? *Commun. ACM*, 35(12) :29–38, 1992.

[Bel89] R. K. Belew. Adaptive information retrieval : using a connectionist representation to retrieve and learn about documents. In *SIGIR '89 : Proceedings of the 12th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 11–20, New York, NY, USA, 1989. ACM Press.

[Bes02] R. Besançon. *Intégration de connaissances syntaxiques et sémantiques dans les répresentations vectorielles de textes*. PhD thesis, Ecole Polytechniques Fédérale de Lausanne, 2002.

[BGB03] J.C Bottraud, G. Bisson G., and M.F. Bruandet. Une aide personnalisée et adaptative pour la recherche d'information sur le web. In *Proceeding of INFORSID (INFormatique des ORganisations et Systèmes d'Information et de Décision)*, pages 235–250, Nancy, 3-6 Juin, 2003.

[BH00] J. Budzik and K.J. Hammond. User interactions with everyday applications as context for just-in-time information access. In *IUI '00 : Proceedings of the 5th international conference on Intelligent user interfaces*, pages 44–51, New York, NY, USA, 2000. ACM Press.

[BM85] D.C. Blair and M.E. Maron. An Evaluation of Retrieval Effectivness for a Full-Text Document Retrieval System. *Communication of the ACM*, 28(3) :289–299, 1985.

[Bot04] J.C Bottraud. *Un assistant adaptatif pour la recherche d'information*. PhD thesis, Université Joseph Fourier - Grenoble I, 2004.

[Bou92] M. Boughanem. *Systèmes de recherche d'information : d'un modele classique à un modèle connexionniste*. PhD thesis, Université Paul Sabatier de Toulouse, 1992.

[Bou00] M. Boughanem. Formalisation et spécification des systèmes de recherche et de filtrage d'information. HDR, Université Paul Sabatier de Toulouse, 2000.

[BRC99] R. Besançon, M. Rajman, and J.C. Chappelier. Textual Similarities based on a Distributional Approach. In *Proceedings of the Tenth International Workshop on Database And Expert Systems Applications (DEXA99), Firenze, Italy*, pages 180–184, 1999.

[BRS00] K. Bradley, R. Rafter, and B. Smyth. Case-based user profiling for content personalisation. In *AH '00 : Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 62–72, London, UK, 2000. Springer-Verlag.

[BSAS94] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using SMART : TREC 3. In *Text REtrieval Conference*, pages 0–, 1994.

[BSMS96] C. Buckley, A. Singhal, M. Mitra, and G. Salton. New retrieval approches using SMART : TREC 4. In *Text REtrieval Conference*, pages 25–48, 1996.

[BT01] M. Boughanem and M. Tmar. Apprentissage incrémental dans un systeme de filtrage adaptatif. In *Proceeding of VSST*, pages 313–319, 2001.

[Cal98] J.P. Callan. Learning while filtering documents. In *SIGIR '98 : Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 224–231. ACM Press, 1998.

[CCCH02] C.J. Crouch, D.B. Crouch, Q. Chen, and S.J. Holtz. Improving the retrieval effectiveness of very short queries. *Inf. Process. Manage.*, 38(1) :1–36, 2002.

[CCG94] W.S. Cooper, A. Chen, and F.C. Gey. Experiments in the probabilistic retrieval of full text documents. In *Proceeding of the third Text REtrieval Conference*, pages 127–134, 1994.

[CCH92] J.P. Callan, W.B. Croft, and S.M. Harding. The INQUERY retrieval system. In *Proceedings of DEXA-92, 3rd International Conference on Database and Expert Systems Applications*, pages 78–83, 1992.

[CGR91] G.A. Carpenter, S. Grossberg, and J.H. Reynolds. Artmap : Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Netw.*, 4(5) :565–588, 1991.

[CH79] W.B. Croft and D.J. Harper. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation*, 35(4) :285–295, 1979.

[Cha90] J. Chauché. Détermination sémantique en analyse structurelle : une expérience basée sur une definition de distance. *TA Information, vol. 2*, pages 121–167, 1990.

[Che04] C.M. Chen. Incremental personalized web page mining utilizing self-organizing hcmac neural network. *Web Intelligence and Agent System*, 2(1) :21–38, 2004.

[CK00] P.M. Chen and F.C. Kuo. An Information Retrieval System Based on User Profile. *The journal of Systems and Software*, 54 :3–8, 2000.

[CLP02] D. Cosley, S. Lawrence, and D.M. Pennock. REFEREE : An open framework for practical testing of recommender systems using researchindex.

In *28th International Conference on Very Large Databases, VLDB 2002*, Hong Kong, August 20–23 2002.

[Coo91] W.S. Cooper. Inconsistencies and misnomers in probabilistic ir. In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Chicago, Illinois, USA, October 13-16, 1991 (Special Issue of the SIGIR Forum)*, pages 57–61. ACM, 1991.

[Cre97] F. Crestani. Comparing neural ad probabilistic relevance feedback in an interactivea information retrieval system. In *Proceeding of the 1994 IEEE Internationale Conference on Neural Networks*, pages 3426–3430, 1997.

[CS98] L. Chen and K. Sycara. WebMate : Personal Agent for Browsing and Searching. In *Proceeding of the Second International Conference on Autonomous Agents*, pages 132–139, 1998.

[CSO96] H. Chen, C. Shuffels, and R. Owings. Internet categorization and search : a self-organizing approach. *Journal of Visual Communication and Image Representation*, pages 88–102, 1996.

[CY92] C.J. Crouch and B. Yang. Experiments in automatic statistical thesaurus construction. In *SIGIR '92 : Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 77–88, New York, NY, USA, 1992. ACM Press.

[Dan94] C. Danilowicz. Modelling of user preferences and needs in boolean retrieval systems. *Information Processing& Management*, 30(3) :363–378, 1994.

[DD80] M. Dillon and J. Desper. The use of automatic relevance feedback in boolean retrieval systems. *Journal of documentation*, 36(3) :197–208, 1980.

[DDL+90] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harsman. Indexing by latent semantic analysis. *Journal of the Society for Information Science, vol. 41*, pages 391–407, 1990.

[DLL96] T. Dumais, T.K. Landauer, and M.L. Littman. Automatic cross-linguistic information retrieval using latent semantic indexing. In *SIGIR' 96 Workshop on Cross-Linguistic Information Retrieval*, 1996.

[DN01] C. Danilowicz and H.C. Nguyen. User Profile in Information Retrieval Systems. In *Proceedings of the 23rd International Scientific School (ISAT 2001)*, pages 117–124. PWr Press, 2001.

[DN02] C. Danilowicz and H.C. Nguyen. Using User Profiles in Intelligent Information Retrieval. In *Proceedings of the 13th International Symposium on Foundations of Intelligent Systems*, pages 223–231. Springer-Verlag, 2002.

[Eft00] E.N. Efthimiadis. Interactive query expansion : a user-based evaluation in a relevance feedback environment. *Journal of the American Society for Information Science*, 51(11) :989–1003, 2000.

[FD92] P.W. Foltz and S.T. Dumais. Personalized information delivery : an analysis of information filtering methods. *Commun. ACM*, 35(12) :51–60, 1992.

[FDD+88] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. Information retrieval using a singular value decomposition model of latent semantic structure. In *SIGIR '88 : Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 465–480, New York, NY, USA, 1988. ACM Press.

[GCP03] S. Gauch, J. Chaffee, and A. Pretschner. Ontology-based personalized search and browsing. *Web Intelli. and Agent Sys.*, 1(3-4) :219–234, 2003.

[GNOT92] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12) :61–70, 1992.

[Har75] S. P. Harter. A probabilistic approach to automatic keyword indexing. *Journal of the American Society for Information Science*, 26(4) :197–206, 1975.

[Hie02] D. Hiemstra. Term-specific smoothing for the language modeling approach to information retrieval : the importance of a query term. In *SIGIR '02 : Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 35–41, New York, NY, USA, 2002. ACM Press.

[HKBR99] J.L. Herlocker, J.A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99 : Pro-*

ceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pages 230–237, New York, NY, USA, 1999. ACM Press.

[HMIH00] K. Hoashi, K. Matsumoto, N. Inoue, and K. Hashimoto. Document filtering method using non-relevant information profile. In *SIGIR '00 : Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 176–183. ACM Press, 2000.

[Hof01] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2) :177–196, 2001.

[Hof04] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1) :89–115, 2004.

[HRZ04] D. Hiemstra, S. Robertson, and H. Zaragoza. Parsimonious language models for information retrieval. In *SIGIR '04 : Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 178–185, New York, NY, USA, 2004. ACM Press.

[Ide71] E. Ide. New experiments in relevance feedback. *G. Salton, editor, The SMART retrieval system*, pages 337–354, 1971.

[JBH97] W.M. Shaw Jr, R. Burgin, and P. Howell. Performance standards and evaluations in ir test collections : Cluster-based retrieval models. *Information Processing & Management*, 33(1) :1–14, 1997.

[JC94] Y. Jing and W.B. Croft. An association thesaurus for information retrieval. In *Proceedings of RIAO-94, 4th International Conference "Recherche d'Information Assistee par Ordinateur"*, pages 146–160, New York, US, 1994.

[JH92] A. Jennings and H. Higuchi. A personal news service based on a user model neural network, 1992.

[JJ70] K. Spark Jones and D.M. Jackson. The use of automatically-obtained keyword classifications for information retrieval. *Information Processing and Management*, 5 :175–201, 1970.

[KGDC00] K. L. Kwok, L. Grunfeld, N. Dinstl, and M. Chan. Trec-9 cross language, web and question-answering track experiments using pircs. In *Proceedings of TREC-9*, 2000.

[KHZ00] Y.H. Kim, S.Y. Hahn, and B.T. Zhang. Text filtering by boosting naive bayes classifiers. In *SIGIR '00 : Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 168–175, New York, NY, USA, 2000. ACM Press.

[KKLH96] T. Kohonen, S. Kasri, K. Lagus, and T. Honkela. Self-organizing maps of document collection : A new approach to interactive exploration. In *Proceedings of the Second International Conference on Knowledge Discovry and Data Mining*, pages 238–243, 1996.

[Koh89] T. Kohonen. Self-organization and associative memory. In *Springer-Verlag*, 1989.

[Kwo89] K.L. Kwok. A neural network for probabilistic information retrieval. In *SIGIR '89 : Proceedings of the 12th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 21–30. ACM Press, 1989.

[Lee01] W.S. Lee. Collaborative learning for recommender systems. In *Proc. 18th International Conf. on Machine Learning*, pages 314–321. Morgan Kaufmann, San Francisco, CA, 2001.

[Lef00] P. Lefèvre. *La Recherche d'informations, du texte intégral au thésaurus.* Hermès, Paris, 2000.

[LF92] A. Lelu and C. Fançois. Information retrieval based on a neural unsupervised extraction of thematic fussy clusters. In *Neuro-Nîmes 92 : Les réseaux neuro-mimétiques et leurs applications*, pages 93–104, 2-6 novembre 1992, Nîmes, France., 1992.

[LG00] S. Lawrence and C.L. Giles. Accessibility of information on the web. *Intelligence*, 11(1) :32–39, 2000.

[Lin97] X. Lin. Map displays for information retrieval. *Journal of American Society for Information Science*, 48(1) :40–54, 1997.

[Luh58] H.P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2 :159–165, 1958.

[LYM04] F. Liu, C. Yu, and W. Meng. Personalized web search for improving retrieval effectiveness. *IEEE Transactions on Knowledge and Data Engineering*, 16(1) :28–40, January 2004.

[LZO03] T. Li, S. Zhu, and M. Ogihara. Efficient multi-way text categorization via generalized discriminant analysis. In *CIKM '03 : Proceedings of the twelfth international conference on Information and knowledge management*, pages 317–324, New York, NY, USA, 2003. ACM Press.

[MADP04] A. Mokrane, R. Arezki, G. Dray, and P. Poncelet. Cartographie Automatique du Contenu d'un Corpus de Documents Textuels. In *Proceeding of the 7th international conference on the statistical analysis of textual data JADT, 12-15 mars 2004, Louvain-La-Neuve, Belgique*, 2004.

[MDLN02] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Discovery and evaluation of aggregate usage profiles for web personalization. *Data Mining and Knowledge Discovery*, 6(1) :61–82, 2002.

[MGT+87] T.W. Malone, K.R. Grant, F.A. Turbak, S.A. Brobst, and M.D. Cohen. Intelligent information-sharing systems. *Commun. ACM*, 30(5) :390–402, 1987.

[MH04] M.R. McLaughlin and J.L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *SIGIR '04 : Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329–336, New York, NY, USA, 2004. ACM Press.

[MK60] M. Maron and J. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM*, 7 :216–243, 1960.

[MK90] S.H. Myaeng and R.R. Korfhage. Integration of user profiles : Models and Experiements in Information Retrieval. *Information Processing & Management*, 26 :719–738, 1990.

[Mob] B. Mobasher. A web personalization engine based on user transaction clustering. *In proceedings of the 9th Workshop on Information Technologies and Systems (WITS'99)*, Decembre.

[Moc96] K. Mock. *Intelligent Information Filtering via Hybrid Techniques : Hill Climbing, Case-based Reasoning, Index Patterns, and Genetic Algorithms.* PhD thesis, Ph.D. thesis, University of California Davis, 1996.

[Mot94] J. Mothe. *Modèle connexionniste pour la recherche d'information.* PhD thesis, Université Paul Sabatier de Toulouse, 1994.

[Mot00] J. Mothe. Recherche et exploration d'information, découverte de connaissances pour l'accès à l'information. HDR, Université Paul Sabatier de Toulouse, 2000.

[Moz99] M.C. Mozer. Inductive information retrieval using parallel distributed computation. In *Technical report, Cognitive science, UCSD, La Jolla, CA*, May 1999.

[MS94] M. McElligot and H. Sorensen. An Evolutionary Connectionist Approach to Personnal Information Filtering. In *Proceedings of the Fourth Irish Neural Network Conference, Dublin, Ireland*, pages 141–146, 1994.

[MTP01] F. Masseglia, M. Teisseire, and P. Poncelet. Real-time web usage mining : A heuristic based distributed miner. In *Proceedings of the Web Information Systems Engineering (WISE'01)*, pages 288–296, 2001.

[MWB94] F. Menczer, W. Willuhn, and R. K. Belew. An endogenous fitness paradigm for adaptive information agents. In *In Proceedings of Third International Conference on Information and Knowledge Management (CIKM'94) - Workshop on Intelligent Information Agents*, 1994.

[MZ97] A. Moukas and G. Zacharia. Evolving a Multi-Agent Information Filetering Solution in Amalthea. In *Proceedings of the First international Conference on Autonomous Agents*, pages 394–403, 1997.

[OMK91] Y. Ogawa, T. Morita, and K. Kobayashi. A fuzzy document retrieval system using the keyword connection matrix and a learning method. *Fuzzy Sets Syst.*, 39(2) :163–179, 1991.

[PB97] M. Pazzani and D. Billisus. Learning and Revising User Profiles : The Identification of Interesting Web Sites. *Machine Learning Journal*, pages 313–331, 1997.

[PC98] J.M. Ponte and W.B. Croft. A language modeling approach to information retrieval. In *SIGIR '98 : Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, USA, 1998. ACM Press.

[PG99] A. Pretschner and S. Gauch. Ontology based personalized search. In *ICTAI*, pages 391–398, 1999.

[Poi99] M. Poinçot. *Classification et recherche d'information bibliographique par l'utilisation des cartes auto-organisatrices, application en astronomie.* PhD thesis, Université Louis Pasteur, strasbourg, 1999.

[QF93] Y. Qiu and H-P Frei. Concept-based query expansion. In *Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval*, pages 160–169, Pittsburgh, US, 1993.

[RBS00] R. Rafter, K. Bradley, and B. Smyth. Personalised retrieval for online recruitment services. In *Proceedings of the 22nd Annual Colloquium on Information Retrieval*, Cambridge, UK, 2000.

[Rij77] C. J. Van Rijsbergen. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation*, 33(2) :106–119, 1977.

[Rij79] C. J. Van Rijsbergen. *Information Retrieval.* Butterworth-Heinemann, 1979.

[RJ76] S. Robertson and K. Spark Jones. Relevance weighting of search terms. *Journal of Americain Society for Information Retrieval*, 27(3) :129–146, 1976.

[RJ97] S. Robertson and K. Jones. Simple proven approaches to text retrieval. Technical Report TR356, Cambridge University Computer Laboratory, 1997.

[Rob77] S. Robertson. The probabilistic ranking principle in IR. *Journal of Documentation*, 33 :294–304, 1977.

[Roc66] J.J. Rocchio. Document retrieval systems : optimization and evaluation, doctoral dissertation, harvard university. *In Report ISR-10, to the national science foundation, Harvard computational laboratory, Cambridge, MA*, 1966.

[Roc71] J.J. Rocchio. Relevance Feedback in Information Retrieval. *In G. Salton, the SMART Retrieval System : Experiments in Automatic Document Processing*, pages 313–323, 1971.

[Rog52] P. Roget's. Roget's international thesaurus. 1852.

[RRP81] S. E. Robertson, C. J. Van Rijsbergen, and M. F. Porter. Probabilistic models of indexing and searching. In *SIGIR '80 : Proceedings of the 3rd annual ACM conference on Research and development in information retrieval*, pages 35–56, Kent, UK, UK, 1981. Butterworth & Co.

[RW94] S. Robertson and S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. *Journal of Documentation*, 33 :294–304, 1994.

[RWHG94] S.E. Robertson, S. Walker, M.M. HancockBeaulieu, and M. Gatford. Okapi at TREC 2. In *TREC-3, the 2nd Text Retrieval Conference*, pages 21–34, 1994.

[RWJ+95] S.E. Robertson, S. Walker, S. Jones, M.M. HancockBeaulieu, and M. Gatford. Okapi at TREC 3. In *In D.Harman, editor, Proc. TREC-3, the 3rd Text Retrieval Conference*, pages 109–127, 1995.

[Sal71] G. Salton. *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[Sal89] G. Salton. *Automatic text processing : the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[SB90] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4) :288–297, 1990.

[SBR02] B. Smyth, K. Bradley, and R. Rafter. Personalization techniques for online recruitment services. *Commun. ACM*, 45(5) :39–40, 2002.

[SFW83] G. Salton, E.A. Fox, and H. Wu. Extended boolean information retrieval. *Commun. ACM*, 26(11) :1022–1036, 1983.

[SG83] G. Salton and M.J Mc Gill. Introduction to Modern Information Retrieval. *New York : McGraw-Hill*, 1983.

[She94] B.D. Sheth. A learning approach to personalized information filtering. Master's thesis, January 1994.

[Sin97] A. Singhal. *Term Weighting Revisited*. PhD thesis, Cornell University, Ithaca, NY, USA, 1997.

[SK92] P. Schauble and D. Knaus. The various roles of information structures. In *In Proceedings of the 16. Jahrestagung der Gesellschaft fur Klassifikation*, pages 282–290, 1992.

[SKKR01] B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th World Wide Web Conference*, pages 285–295, 2001.

[SSMB95] A. Singhal, G. Salton, M. Mitra, and C. Buckley. Document Length Normalization. Technical report, Departement of Computer Science, Cornell University, 1995.

[SSS98] R.E. Schapire, Y. Singer, and A. Singhal. Boosting and rocchio applied to text filtering. In *SIGIR '98 : Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 215–223. ACM Press, 1998.

[SVF84] G. Salton, C. E. Voorhees, and E.A Fox. A comparison of two methods for boolean query relevance feedback. *Information processing&Management*, 20 :200–210, 1984.

[SYY75] G. Salton, C. Yang, and C. Yu. A Theory of term Importance in Automatic Text Analysis. *Information Processing and Management*, 24 :513–523, 1975.

[TC91] H. Turtle and W.B. Croft. Evaluation of an inference network-based retrieval model. *ACM Trans. Inf. Syst.*, 9(3) :187–222, 1991.

[Teb04] H. Tebri. *Formalisation et spécification d'un système de filtrage incrémental d'information.* PhD thesis, Université Paul Sabatier de Toulouse, 2004.

[TS92] J. Tague-Sutcliffe. The pragmatics of information retrieval experimentation, revisited. *Inf. Process. Manage.*, 28(4) :467–490, 1992.

[TT98] A. Tan and C. Teo. Learning User Profiles for Personalized Information Dissemination. In *Proceedings, 1998 IEEE International Joint Conference on Neural Networks, Alaska*, pages 183–188, 1998.

[VH99] E. Voorhees and D. Harman. Evaluation tecxhniques and measures - appendix a. In *Proceedings of the Eighth Text REtrieval Conference*, 1999.

[VRJ03] O. Vechtomova, S. Robertson, and S. Jones. Query expansion with long-span collocates. *Information Retrieval*, 2(6) :251–273, 2003.

[WCY93] S.K.M. Wong, Y.J. Cai, and Y.Y. Yao. Computation of term associations by a neural network. In *SIGIR '93 : Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 107–115, New York, NY, USA, 1993. ACM Press.

[WH91] R. Wilkinson and P. Hingston. Using the cosine measure in a neural network for document retrieval. In *SIGIR '91 : Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 202–210. ACM Press, 1991.

[WIY99] D.H. Widyantoro, T.R Ioerger, and J. Yen. An Adaptative Algorithm for Learning Changes in User Interests. In *Proceedings of the Eighth International Conference on Information and Knowledge Management*, pages 405–412, 1999.

[WIY01] D.H. Widyantoro, T.R Ioerger, and J. Yen. Learning User Interest Dynamics with a Three-Descriptor Representation. *Journal of the American Society of Information Science*, 52(3) :212–225, 2001.

[WIY03] D.H. Widyantoro, T.R. Ioerger, and J. Yen. Tracking Changes in User Interests with a Few Relevance Judgments. In *Proceedings of the International Conference on Information and Knowledge Management CIKM'03*, 2003.

[WPW95a] E. Weiner, J. O. Pedersen, and A. S. Weigend. A neural network approach to topic spotting. In *In Proceedings of SDAIR'95*, pages 317–332, 1995.

[WPW95b] Erik D. Wiener, Jan O. Pedersen, and Andreas S. Weigend. A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 317–332, Las Vegas, US, 1995.

[WV05] Y. Wang and O. Vechtomova. Exploring the use of term proximity in collocate-ranking for query expansion. In *In Proceedings of the Joint ACH/ALLC (Association for Computers and the Humanities/Association for Literary and Linguistic Computing) Conference*, 2005.

[WZW85] S. K. M. Wong, W. Ziarko, and P.C.N. Wong. Generalized vector spaces model in information retrieval. In *SIGIR '85 : Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 18–25, New York, NY, USA, 1985. ACM Press.

[XBC94] J. Xu, J. Broglio, and B. Croft. The design and implementation of a part of speech tagger for english. Technical report, University of Massachusetts, Amherst, MA, USA, 1994.

[XC96] J. Xu and W.B. Croft. Query expansion using local and global document analysis. In *SIGIR '96 : Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 4–11, New York, NY, USA, 1996. ACM Press.

[XC00] J. Xu and W.B Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Trans. Inf. Syst.*, 18(1) :79–112, 2000.

[Xu97] J. Xu. *Solving the word mismatch problem through automatic text analysis*. PhD thesis, IUniversity of Massachusetts, 1997.

[Zad65] L. Zadeh. Fuzzy sets. *Information and Control*, 8 :338–353, 1965.

[ZH01] S. Zelikovitz and H. Hirsh. Using lsi for text classification in the presence of background text. In *CIKM '01 : Proceedings of the tenth international conference on Information and knowledge management*, pages 113–118, New York, NY, USA, 2001. ACM Press.

[Zip49] G. K. Zipf. *Human Behavior and the Principle of Least-Effort*. Addison-Wesley, Cambridge, MA, 1949.