Diffusion :

- M.Fofana
- documents du projet

date de diffusion(s) :15.10.02

Dernière impression le : 02/02/03 15:47

Académie de Montpellier

Université de Montpellier II

Sciences et Techniques du Languedoc

Diplôme d'Etudes Approfondies « Informatique »

Implémentation d'une capacité d'apprentissage à un centre d'information sur le secteur des pêches en guinée

2ème Partie

Auteur : Cheikh Ba.

Tuteur : Philippe Reitz.

Sommaire

Introduction

L'explosion du nombre de sites webs connectés sur Internet et la croissance accélérée du nombre d'internautes confirment de plus en plus la position du web comme un média de masse. Par conséquent, le problème de « l'audience » des sites web revêt de plus en plus d'importance. L'audience d'un site web est un subtil mélange entre le nombre de visiteurs de ce site(le quantitatif) et l'intérêt qu'ont les internautes à visiter le site(le qualitatif). Dans ce contexte, de nombreux travaux se sont intéressés à étudier les problématiques de l'auto adaptation des sites web et de la classification des internautes.

L'application des techniques du *data mining* au web appelée *web data mining* **[1]** est devenue le centre d'intérêt d'un nombre grandissant de chercheurs.

Il y a actuellement dans le *web data mining* trois principales directions de recherche : (i) recherche d'informations, (ii) analyse de la structure de liens du web, (iii) analyse des comportements utilisateurs. Ce dernier, qui présente apparemment beaucoup plus d'intérêts, est axé sur des techniques qui étudient les modèles de navigation des utilisateurs, permettant de personnaliser la présentation pour l'utilisateur individuel, et d'améliorer la structure du site selon les types d'utilisateurs.

Ce présent rapport est fourni en guise de complément au premier qui, dû à un problème de temps, n'avait pas fait l'objet de recherches dans le domaine en question.

Ainsi, y seront présentés : une vue globale sur la problématique du *web data mining*, une classification des approches d'adaptation, des méthodologies et algorithmes (état de l'art) et une brève discussion sur le centre d'information en guise de conclusion.

I. Le *web usage mining*

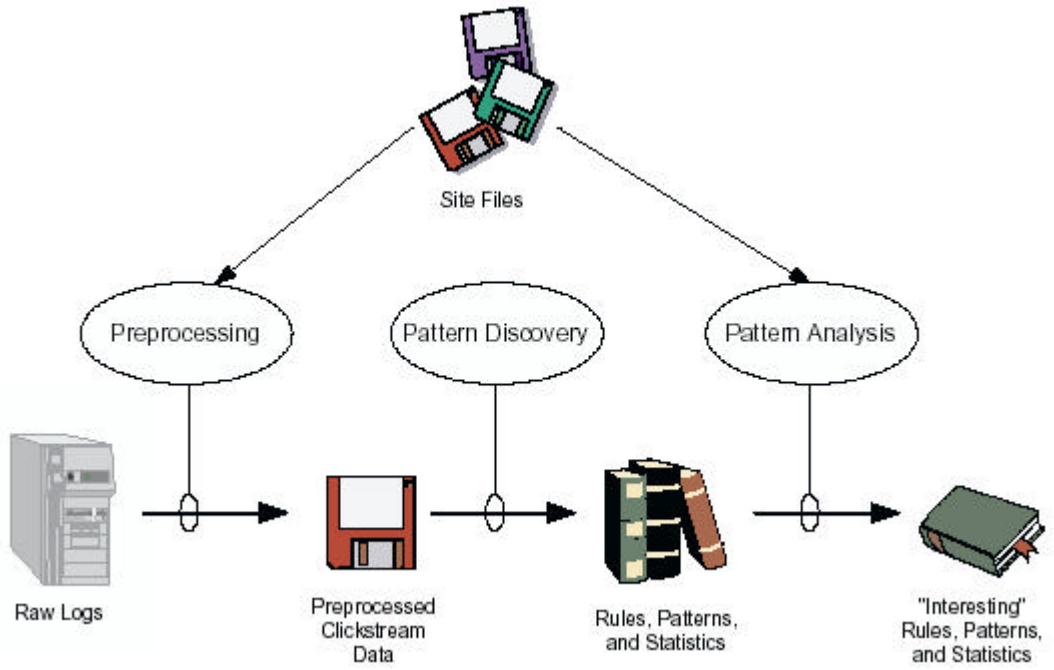
Le *web usage mining* est l'application des techniques du *data mining* pour découvrir les modèles d'utilisation à partir des données d'accès, afin de comprendre les utilisateurs et leur faciliter la tâche.

Une des étapes clef dans l'extraction de connaissances dans les bases de données est la création de l'ensemble de données adéquat pour l'analyse. Il y a plusieurs types de données pouvant être utilisées dans le *web mining*. Dans **[3]**, de telles données sont classifiées parmi les types suivants :

- contenu :ce sont les «vraies » données dans les pages web, c'est à dire les données qui devaient être transmises à l'utilisateur. Il s'agit souvent (mais pas seulement) de données textuelles et d'images.
- structure :donnés décrivant l'organisation du contenu. L'information sur la structure de l'intra-page comprend les arrangements des différentes balises HTML ou XML. Ceci peut être représenté par une structure d'arbre où la balise <HTML> devient la racine.

- utilisation :données décrivant les modèles d'utilisation des pages web à l'exemple des adresses IP, des références sur les pages, des dates et heures d'accès.
- profil utilisateur : données qui fournissent des informations géographiques sur les utilisateurs du site.

Il y a trois phases **[7]** dans le processus du *web mining* :le prétraitement(*preprocess*), la découverte de modèles de navigation(*pattern discovery*) et l'analyse de ces modèles(voir figure suivante).



II.1. Le preprocessing

Le *preprocessing* ou prétraitement consiste à convertir les informations sur l'utilisation, le contenu et la structure du contenu dans les sources de données disponibles en des abstractions de données nécessaires pour la découverte de modèles.

II.1.1. Prétraitement de l'utilisation

Le prétraitement de l'utilisation(ou *usage preprocessing*) est la tâche la plus difficile dans le processus du *web usage mining*, à cause de la nature incomplète des données disponibles. Seuls (ou presque) l'adresse IP et l'agent de navigation sont utilisés pour identifier l'utilisateur et les sessions. Ainsi quelques problèmes peuvent surgir :

- unique adresse IP/plusieurs sessions – les fournisseurs d'accès ont souvent un groupe de serveurs proxy par lesquels les utilisateurs accèdent à internet. Un unique serveur proxy peut avoir différents utilisateurs accédant au site web pratiquement à la même période.
- plusieurs adresses IP/unique session – quelques fournisseurs d'accès assignent au hasard chaque requête utilisateur à un ou plusieurs adresses IP. Dans ce cas, une unique session peut avoir plusieurs adresses IP.

- plusieurs adresses IP/unique utilisateur – un utilisateur qui accède au web à partir de différentes machines aura des adresses IP différentes d'une session à l'autre.
- plusieurs agents/utilisateur unique – encore, un internaute utilisant plus d'un navigateur, même sur une même machine, apparaîtra (pour certaines approches) sous la forme de plusieurs utilisateurs.

En supposant que chaque utilisateur a maintenant été identifié, ses visite doivent être découpées en sessions. Dans la mesure où les pages demandées sur les autres serveurs ne sont pas disponibles, il est difficile de savoir quand est ce qu'un utilisateur a réellement quitté le site web. Un délai de 30 minutes est souvent utilisé comme délimiteur d'une visite en plusieurs sessions. Ce délai est issu des résultats de **[11]**.

La figure suivante montre un échantillon de données d'accès, illustrant les différents problèmes évoqués plus haut.

#	IP Address	Userid	Time	Method/ URL/ Protocol	Status	Size	Referrer	Agent
1	123.456.78.9	-	[25/Apr/1998:03:04:41 -0500]	"GET A.html HTTP/1.0"	200	3290	-	Mozilla/3.04 (Win95, I)
2	123.456.78.9	-	[25/Apr/1998:03:05:34 -0500]	"GET B.html HTTP/1.0"	200	2050	A.html	Mozilla/3.04 (Win95, I)
3	123.456.78.9	-	[25/Apr/1998:03:05:39 -0500]	"GET L.html HTTP/1.0"	200	4130	-	Mozilla/3.04 (Win95, I)
4	123.456.78.9	-	[25/Apr/1998:03:06:02 -0500]	"GET F.html HTTP/1.0"	200	5096	B.html	Mozilla/3.04 (Win95, I)
5	123.456.78.9	-	[25/Apr/1998:03:06:58 -0500]	"GET A.html HTTP/1.0"	200	3290	-	Mozilla/3.01 (X11, I, IRIX6.2, IP22)
6	123.456.78.9	-	[25/Apr/1998:03:07:42 -0500]	"GET B.html HTTP/1.0"	200	2050	A.html	Mozilla/3.01 (X11, I, IRIX6.2, IP22)
7	123.456.78.9	-	[25/Apr/1998:03:07:55 -0500]	"GET R.html HTTP/1.0"	200	8140	L.html	Mozilla/3.04 (Win95, I)
8	123.456.78.9	-	[25/Apr/1998:03:09:50 -0500]	"GET C.html HTTP/1.0"	200	1820	A.html	Mozilla/3.01 (X11, I, IRIX6.2, IP22)
9	123.456.78.9	-	[25/Apr/1998:03:10:02 -0500]	"GET O.html HTTP/1.0"	200	2270	F.html	Mozilla/3.04 (Win95, I)
10	123.456.78.9	-	[25/Apr/1998:03:10:45 -0500]	"GET J.html HTTP/1.0"	200	9430	C.html	Mozilla/3.01 (X11, I, IRIX6.2, IP22)
11	123.456.78.9	-	[25/Apr/1998:03:12:23 -0500]	"GET G.html HTTP/1.0"	200	7220	B.html	Mozilla/3.04 (Win95, I)
12	209.456.78.2	-	[25/Apr/1998:05:05:22 -0500]	"GET A.html HTTP/1.0"	200	3290	-	Mozilla/3.04 (Win95, I)
13	209.456.78.3	-	[25/Apr/1998:05:06:03 -0500]	"GET D.html HTTP/1.0"	200	1680	A.html	Mozilla/3.04 (Win95, I)

l'adresse IP 123.456.78.9 est le responsable de 3 sessions, les adresses IP 209.456.78.2 et 209.456.78.3 sont responsables d'une quatrième session. En combinant certaines informations notamment sur les agents, on peut subdiviser les enregistrements de la ligne 1 à 11 en 3 sessions(A-B-F-O-G, L-R et A-B-C-J). Sans utiliser les cookies, l'id encastré de session ou une méthode de collection de données coté client, il n'y a pas de moyen de déterminer que les lignes 12 et 13 sont réellement une session unique.

II.1.2. Prétraitement du contenu :

Le prétraitement du contenu consiste en la conversion du texte, des images, des scripts, et d'autres fichiers sous une forme utilisable par le processus du *web data mining*. Cela consiste souvent à effectuer une analyse de contenu comme une classification ou un regroupement.

Pour exécuter les algorithmes d'analyse de contenu, les informations devront d'abord être converties en un format quantifiable. Une version sur le modèle des espaces vectoriels peut être utilisée cette tâche. Les textes peuvent être transformés en vecteurs de mots.

Concernant les images par exemple, des mots clef ou des textes descriptifs pourront les substituer.

II.1.3. Prétraitement de la structure :

La structure d'une page est créée par les liens hypertextes entre les pages. La structure peut être obtenue et traitée de la même manière que le contenu d'un site. Les contenus dynamiques(et donc les liens) posent beaucoup plus de problèmes que les pages statiques.

II.2. Découverte des modèles :

La découverte des modèles de navigation concerne plusieurs méthodes et algorithmes provenant de différentes branches de recherche telles les statistiques, le *data mining* et le *machine learning*. Cependant cette section n'a pas pour intention de décrire tous les algorithmes et techniques dérivant de ces différentes branches. Elle présente plutôt les types d'analyse qui ont été appliqués au domaine du web. Les méthodes développées à partir des différentes branches de recherche doivent pendre en considération les différents types d'abstraction de données et les connaissances disponibles pour le *web mining*.

Par exemple, dans la recherche de règles d'associations, la notion de transaction dans l'analyse du problème « market-basket » ne prend pas en considération l'ordre dans lequel les items sont sélectionnés. Par contre dans le *web usage mining*, une session est une séquence ordonnée de pages demandées par un utilisateur.

II.2.1. Analyse statistique :

La technique statistique est la méthode la plus commune pour extraire des connaissances à propos des visiteurs d'un site web. En analysant le fichier session, on peut calculer plusieurs types de grandeurs en statistique descriptive(fréquences, moyennes, médiane…) sur les variables telles que les visites des pages, les temps de visites et les longueurs des chemins parcourus.

En dépit du manque de «profondeur » de ces analyses, ces types de connaissances peuvent être potentiellement utiles pour l'amélioration de la performance du système par exemple.

II.2.2. Règles d'association :

La génération de règles d'association peut être utilisée pour renseigner sur les pages qui sont le plus souvent invoquées ensemble dans une même session.

Dans le contexte du *web usage mining*, les règles d'association se réfèrent aux ensembles de pages accédées ensemble avec une valeur support dépassant un certain seuil prédéfini. Il est à noter que ces pages peuvent ne pas être directement reliées par des hyperliens.

A coté de l'application dans le marketing, la présence ou l'absence de telles règles peut aider les concepteurs de pages web à restructurer leurs sites.

II.2.3. Clustering :

Le *clustering* est une technique regroupant des items ayant des caractéristiques similaires. Dans le domaine du *web usage mining*, il y a deux types de clusters à

découvrir :les clusters d'utilisateurs et les clusters de pages. Le clustering des utilisateurs tend à trouver des groupes d'internautes exhibant des modèles de navigation similaires. D'autre part, le clustering des pages regroupera les pages dont les contenus sont sémantiquement proches.

II.2.4. Classification :

La classification est la tâche consistant à *mapper* un item parmi une ou plusieurs classes prédéfinies.

La classification peut être faite en utilisant des algorithmes d'apprentissage inductif comme les arbres de décision.

Par exemple, la classification au niveau d'un certain site peut amener à découvrir d'intéressantes règles du genre : 30% des utilisateurs achetant en ligne un CD R&B appartiennent au groupe 18-25 ans et habitent la Côte Ouest.

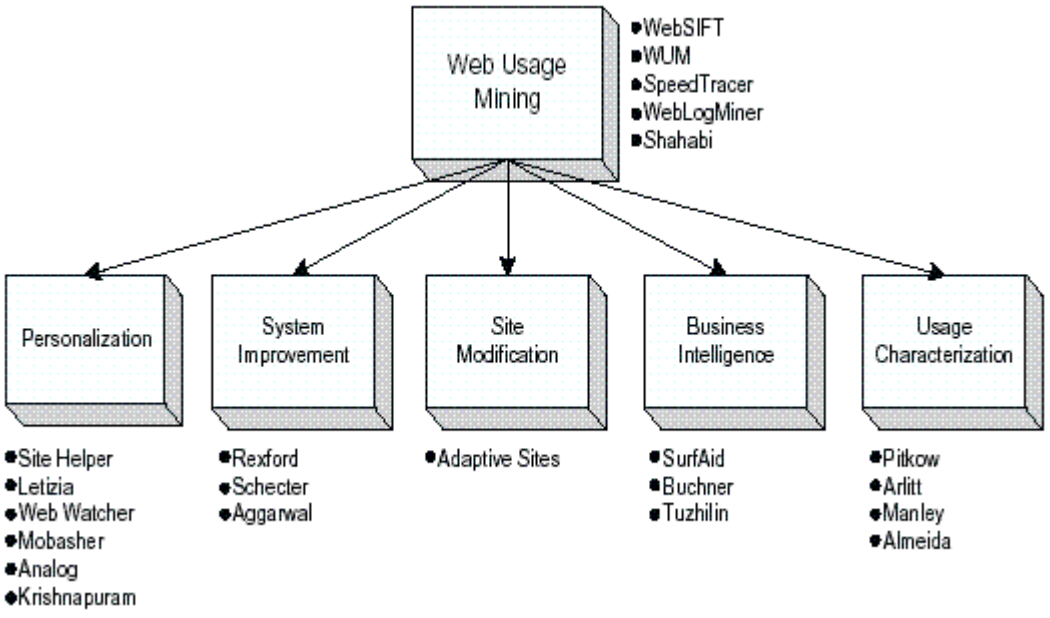
II.2.5. Motifs séquentiels :

La technique sur la découverte des motifs séquentiels consiste à trouver des modèles de sessions tels que la présence d'un ensemble d'items soit suivie par un autre item dans un ensemble ordonné de sessions ou d'épisodes. En utilisant cette approche, les *webmarketer* peuvent prédire les modèles des visites futures qui permettront par exemple de mettre des avertissements visant un certain groupe d'utilisateurs.

II.3. Analyse des modèles :

L'analyse des modèles de navigation est la dernière étape dans tout le processus du *web usage mining*. Il s'agit ici de voir comment exploiter toutes les informations qui ont été obtenues.

Depuis 1996, il y a eu plusieurs projets de recherche et produits commerciaux qui ont eu pour but d'analyser les données d'utilisation du web(voir figure suivante).



III. Classification des approches d'adaptation

Une nouvelle classification des approches d'auto-adaptation des sites Web est proposée dans **[5]**. Les critères de classification les plus récurrents sont : 1) la nature de l'adaptation qui peut porter sur la modification du contenu du site, des liens que contient le site ou la présentation du site, et 2) l'objectif de l'adaptation qui peut être la personnalisation du site ou la transformation du site. La majorité de travaux dans le domaine de l'auto-adaptation des sites Web aborde le problème de l'adaptation du seul point de vue de l'utilisateur. Cependant d'autres points de vues peuvent motiver le recours à l'adaptation des sites Web. Ainsi, et afin de compléter et d'enrichir les schémas de classification existants, il est proposé dans la suite une nouvelle classification fondée sur les cinq critères suivants :

III.1.La nature de l'adaptation :

Nous distinguons entre quatre types d'opérations d'adaptation qui sont les suivantes :

- Adaptation du contenu : Il s'agit de modifier le contenu sémantique du site en fonction de l'utilisation du site. Une autre forme d'adaptation du contenu consiste à insérer des publicités ciblées dans les pages consultées par les utilisateurs. Les publicités peuvent être déterminées en fonction du comportement et/ou du profil de l'utilisateur.
- Adaptation de la structure : Il s'agit de modifier les liens qui relient les différentes pages d'un site Web. L'adaptation structurelle peut se présenter sous la forme d'ajout de nouveaux liens à une page comme c'est le cas dans la plupart des systèmes d'aide à la navigation sur le Web. Une autre forme d'adaptation structurelle consiste à *cacher* des liens existants afin de réduire la complexité de l'espace navigable. D'autres formes consistent à *effacer* des liens existants ou à *inhiber* certains liens. Une dernière forme d'adaptation structurelle consiste à ajouter au site des pages index.
- Adaptation de la présentation : Il s'agit de modifier la présentation du site en termes de couleur, de format et de types de caractères employés. Une forme particulière de ce type d'adaptation concerne l'adaptation de la présentation des liens. Certains systèmes proposent *d'annoter* les liens afin d'indiquer les liens les plus intéressants à suivre.
- Adaptation du support : Il s'agit de modifier le type du support des informations véhiculées par le site Web. Des exemples consistent à modifier la résolution des images contenues dans les pages en fonction de la charge du site ou de modifier le schéma de codage des médias contenus en fonction des caractéristiques techniques de la station de l'utilisateur ou en fonction de certains handicaps de celui-ci. Certains sites changent la langue de présentation du site (anglais, français, etc.) en fonction de l'identité du domaine Internet de rattachement de la station client (livrer

une version française pour les clients dont les adresses IP sont dans le domaine «*.fr* »

III.2. Point de vue de l'adaptation :

Nous distinguons au moins entre quatre points de vues différentes qui peuvent guider un processus d'adaptation. Ces points de vues correspondent aux quatre types d'acteurs qu'on peut identifier dans tout système d'information : l'utilisateur, le concepteur, l'opérateur et le financier. Chacun de ces acteurs a ses propres intérêts dans l'utilisation d'un site qui sont différentes des intérêts des autres. La plupart des systèmes existants adoptent le point de vue de l'utilisateur. Leur objectif est de rapprocher et de faciliter l'accès à l'information pertinente aux utilisateurs. L'insertion dynamique des messages publicitaires dans un site Web correspond à une approche guidée par le point de vue du financier qui cherche, traditionnellement, à augmenter les profits du système.

III.3. La portée de l'adaptation :

Nous distinguons entre deux types de portée d'adaptation, la *personnalisation*

du site et la *transformation* du site. Dans le premier cas il s'agit de modifier le site différemment pour chaque utilisateur et ceci en prenant en compte des caractéristiques spécifiques à cet utilisateur. Des exemples de caractéristiques utilisateurs sont des profils utilisateurs, le comportement utilisateur ou simplement un ensemble de règles

d'adaptation éditées par chaque utilisateur. Dans le deuxième cas il s'agit de modifier le site d'une manière uniforme pour l'ensemble de ses utilisateurs.

III.4. L'apprentissage :

Il s'agit ici de l'utilisation des techniques d'apprentissage automatique par les systèmes d'auto-adaptation. Nous distinguons entre quatre cas possibles. Le cas limite où il n'y pas d'apprentissage, l'apprentissage centré sur un seul utilisateur, l'apprentissage à partir d'un groupe d'utilisateurs et *le méta-apprentissage*.

Ce dernier cas correspond à l'utilisation des techniques de l'apprentissage symbolique pour

faire évoluer les techniques d'adaptation utilisées par le système. Les systèmes fondés sur la construction automatique de profil utilisateur pour guider le processus de l'adaptation sont des exemples de systèmes de deuxième type. L'algorithme *PageGather* (section suivante) est un exemple de système qui met à profit l'expérience d'un groupe d'utilisateurs pour guider le processus de l'adaptation.

III.5. Adaptation «en ligne » contre adaptation « hors ligne » :

Ce critère décrit si l'adaptation du site est faite pendant la navigation des utilisateurs dans le site ou bien fait par lots hors ligne.

IV. Méthodologies et algorithmes

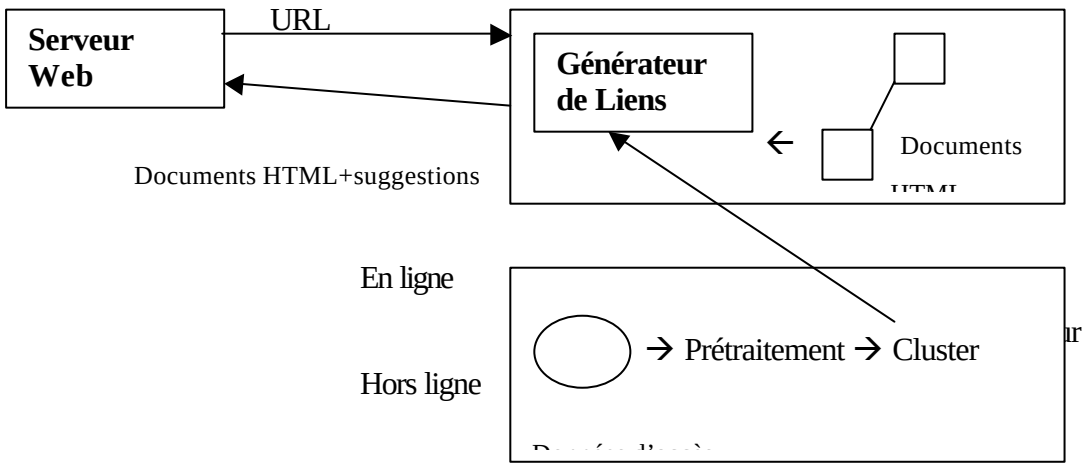
Dans cette section sont décrits quelques algorithmes et méthodologies de l'état de l'art sur le *web mining*.

IV.1. Algorithme *Leader* :classification des visiteurs

Dans **[2]** est décrite une approche pour une classification automatique des visiteurs d'un site web en fonction de leurs modèles de navigation. Les accès utilisateurs sont examinés afin de découvrir des groupes d'utilisateurs exhibant des besoins similaires en information ; c'est à dire accédant à des pages similaires. Ceci peut aboutir à une meilleure compréhension de la manière dont les utilisateurs visitent le site, et même à une amélioration de l'organisation des liens pour une navigation plus commode. Plus intéressant encore, en fonction de la catégorie à laquelle l'utilisateur appartient, il peut y avoir des suggestions de liens pour lui faciliter sa navigation.

IV.1.1. conception du système :

Dans cette section est donnée une vue d'ensemble de cette approche



Le système comprend trois principales composantes : un serveur web capable de stocker les informations sur les sessions utilisateurs, un module hors ligne responsable de l'analyse des accès, et un module en ligne se chargeant de la génération de liens dynamiques.

Dans le module hors ligne, le préprocesseur extrait périodiquement des informations à partir des fichiers d'accès utilisateurs pour générer des enregistrements de sessions utilisateurs. Un enregistrement est généré pour chaque session. Les enregistrements seront par la suite regroupés en catégories, contenant des session similaires.

Le module en ligne effectue la génération dynamique de liens. Quand un utilisateur demande une nouvelle page, ce module essaie de classifier la session partielle courante parmi une ou plusieurs catégories déjà obtenues. La catégorie qui correspond le mieux est identifiée, et les liens vers les pages inexplorées dans ces catégories sont insérés au début de la page à retourner à l'utilisateur.

IV.1.2. le prétraitement :

On peut voir un site web comme étant un ensemble d'items intéressants. Une alternative serait de grouper les pages selon une considération sémantique.

Lors d'une session, un utilisateur peut montrer différents degrés d'intérêt dans ces items. S'il y a *n* items d'intérêts dans un site web, une session utilisateur pourra être représentée comme un vecteur en *n* dimensions, le $i^{ème}$ élément étant le poids, ou le degré d'intérêt assigné à l'ième item. Quand on considère une page web comme un item d'intérêt, on peut lui donner un poids égal au nombre au nombre de fois qu'elle a été visitée, ou à une estimation du temps passé par l'utilisateur sur cette page(peut être normalisé par la longueur de la page), ou au nombre de liens de la page sur lesquels l'utilisateur a cliqué.

La principale tâche du prétraitement est de transformer l'information contenue dans les fichiers d'accès en une représentation vectorielle. L'ordre d'accès aux pages est une partie très importante de l'information, mais n'est pas capturé par cette représentation vectorielle.

IV.1.3. le regroupement :

Une fois que les sessions sont représentées dans un format vectoriel, nous sommes prêts à appliquer un algorithme de regroupement sur elles. Le but dans ce traitement est de découvrir les groupes de sessions exhibant des intérêts similaires. Cela revient au même en trouvant les groupes de vecteurs « similaires ». La similarité peut être définie de plusieurs sortes. Par exemple deux vecteurs sont similaires si la distance euclidienne entre eux est suffisamment petite, ou si l'angle entre eux est suffisamment petit.

Les techniques de regroupement(clustering) sont très étudiées et il y a beaucoup d'algorithmes très connus(leader, k-means, ..). Dans certains algorithmes, un vecteur peut appartenir à plus d'un groupe(cluster).

On peut imposer quelques contraintes par soucis de performance(temps de regroupement). La première est que nous devons nous intéresser qu'aux seules sessions accédant à plus d'un certain nombre de pages appelé *MinNumPages*. Par exemple, il n'est pas utile de regrouper des utilisateurs qui ont juste visité la page d'accueil. Avec cette contrainte, le nombre de sessions pour l'analyse est réduit. En second lieu, nous ne devons considérer que les clusters dépassant une certaine taille appelée *MinClusterSize*. Ceci élimine les clusters insignifiants et peut améliorer la performance.

Cette discussion peut être illustrée avec un algorithme simple, l'algorithme *Leader*.

En entrée nous avons un ensemble *V* de vecteurs.

En sortie nous avons un ensemble *C* de clusters (un cluster est un groupe de vecteurs).

Nous commençons sans cluster et parcourons un à un les vecteurs en entrée. Pour chaque vecteur nous essayons de l'ajouter au cluster le plus connexe tel que la médiane sera inférieure à une distance euclidienne *MaxDistance*. Si ce cluster n'existe pas, le vecteur forme un nouveau cluster.

Mettre C à vide

Pour chaque v

Si

le cardinal de v est supérieur à MinNumPages

Alors

Trouver le cluster c de C tel que la distance entre la médiane de c et v est le minimum(soit d ce minimum) parmi tous les clusters dans C

Si la distance d est inférieure à MaxDistance

Alors

ajouter v à c

Sinon

ajouter {v} à C

Pour chaque c dans C

Si

la taille de c est inférieure à MinClusterSize

Alors

Enlever c de C

Retourner C

Cet algorithme a plusieurs inconvénients. Le plus visible est le fait qu'il ne soit pas invariant aux permutations dans les vecteurs. En plus la distance entre le vecteur et la médiane du cluster auquel il appartient est illimitée.

Toutefois une force importante de cet algorithme est sa rapidité.

Après la découverte des clusters, on peut calculer la médiane de chaque cluster et caractériser ce qu'il représente. Les pages dominantes sont celles auxquelles sont associés les plus grands poids, et on peut donc dire quelles pages caractérisent le cluster.

IV.1.4. Génération dynamiques de liens

Puisqu'un utilisateur navigue à travers les pages d'un site web, nous aurons besoin de suivre ses traces sur les pages visitées, et de le mettre si possible dans une ou plusieurs catégories connues. Nous pourrons donc lui insérer des liens vers des pages intéressantes à suivre.

Pour maintenir l'information sur la session utilisateur active, les accès sont temporairement stockés dans un buffer.

Quand un utilisateur en ligne demande un nouvel URL, le vecteur est mis à jour. Il faut noter qu'à ce stade, seul le vecteur représente un enregistrement partiel de la session en cours. En classifiant le vecteur de la session partielle, la distance entre la médiane d'un cluster et le vecteur partiel peut ne pas être la bonne mesure, puisque le vecteur partiel a

beaucoup plus d'élément nuls. Une alternative est de compter le nombre de pages que l'utilisateur a accédé dans chaque catégorie ;si le décompte est supérieur à un certain seuil prédéfini, la catégorie correspondante est trouvée.

Après que les catégories correspondantes sont identifiées, on va maintenant s'intéresser aux pages dans cette catégorie. Les pages auxquelles l'utilisateur n'a pas encore accédé, et qui sont inaccessibles à partir de l'URL demandée, sont incluses comme suggestion au début du document html retourné au navigateur.

IV.2. Algorithme *page gather* :génération de la page index

Dans **[4]** est présentée une approche utilisant l'algorithme *page gather*(qui met à profit l'expérience d'un groupe d'utilisateurs pour guider le processus d'adaptation), pour tenter de résoudre une partie du problème « *index page synthesis* ».

De quoi s'agit-t-il ?

« *page synthesis* » est la création automatique d'une page web. Un *index page* est une page contenant des liens sur un ensemble de pages sémantiquement proches.

Etant donné un site web et un ensemble d'accès utilisateurs ;il s'agira dans le problème « *index page synthesis* »de créer de nouvelles pages index contenant des liens vers des pages apparentées, mais qui n'ont été liées en aucun moment dans le site. Le fichier des logs est un document contenant une entrée par page demandée au serveur web. Chaque requête renseigne au minimum sur son origine(adresse IP), l'URL demandée et l'heure à laquelle la demande a été faite.

Deux pages sont dites liées s'il existe un lien de l'une vers l'autre ou s'il existe une autre page se liant au deux.

Le problème peut être décomposé en plusieurs sous problèmes :

- quel sera le contenu de la page index? (en terme de liens)
- comment ces liens seront-t-ils ordonnés ?
- comment seront-t-ils étiquetés?
- quel sera le titre de la page ? correspondra-t-il à un concept cohérent ?
- sera-t-il adéquat d'ajouter la page au site? si oui, où l'insérer ?

L'attention n'est ici portée que sur le premier sous problème, à savoir la génération du contenu de la nouvelle page.

IV.2.1. Page Gather :

Etant donné un grand nombre d'accès, la tâche est de trouver des ensembles de pages qui tendent à être cooccurrentes. Le *clustering* est une technique à considérer pour ce genre de problème. Dans le *clustering*(ou regroupement), les données sont représentées dans un espace à N dimensions(comme des vecteurs de mots par exemple). Techniquement parlant, un cluster(groupe) est un ensemble connexe de données relativement « distant » des autres clusters.

Les algorithmes standards de regroupement partitionnent les données en un ensemble de clusters qui s'excluent l'un à l'autre.

Cluster mining(ou analyse de clusters) est une variante du clustering traditionnel qui convient bien dans ce problème. En effet, il s'est avéré plus judicieux d'essayer de trouver un petit nombre de groupements très significatifs, plutôt que de tenter de partitionner la totalité des données. En outre, là où le clustering traditionnel tente de placer chaque donnée(vecteur) dans exactement un cluster, il se peut qu'il y ait dans le *cluster mining*, un même vecteur appartenant à plusieurs clusters.

L'algorithme *page gather* utilise « cluster mining » pour trouver des ensembles de pages apparentées d'un site web.

Par essence, *page gather* prend en entrée les accès et les transforme en une structure de données convenable pour le *cluetering* ; il lui applique ensuite *cluster mining* et donne en sortie des pages index candidates. L'algorithme a cinq principales étapes :

- ➔ traitement des fichiers logs des visiteurs
- ➔ calculer les fréquences de cooccurrence entre les pages et créer une matrice de similitude.
- ➔ créer le graphe correspondant à la matrice, et trouver les cliques maximales(ou les composants connexes) dans le graphe
- ➔ ranger les groupes trouvés et en choisir les meilleurs
- ➔ pour chaque groupe, créer une page web contenant des liens vers les éléments du groupe, la présenter au webmaster pour évaluation.

IV.2.2. Détail des étapes :

- traitement des fichiers log :

Une visite est une séquence ordonnée de pages accédées par un menu utilisateur au cours d'une même session. Cependant, le fichier log est une séquence de pages visitées ou demandées au serveur web. Chaque requête contient l'heure à laquelle elle a été faite, l'URL demandé et l'IP de la machine cliente. On suppose que chaque machine cliente correspond à un seul visiteur.

- Calcul des fréquences de cooccurrence entre les pages :

Pour chaque couple de pages p1 et p2, on calcule **P**(p1/p2), la probabilité pour qu'un visiteur visite p1 sachant qu'il a déjà visité p2. La fréquence de cooccurrence entre p1 et p2 est **F** = min { P(p1/p2) , P(p2/p1) }. Le choix du minimum des probabilités conditionnelles a pour but d'éviter de faire une erreur sur une relation asymétrique pour un vrai cas de similitude. Par exemple une page p1 très sollicitée peut se trouver sur le même chemin d'accès qu'une page inconnue p2. Dans ce cas P(p1/p2) sera très grande, nous amenant à penser que ces pages sont très liées. Par contre P(p2/p1) peut être très petite.

En rappel il a été dit que le but est de trouver des regroupements de pages apparentées, mais n'ayant en aucun moment été liées dans le site. Pour cette raison, il est à éviter des groupements de pages ayant déjà été liées, en mettant un zéro dans chaque case de la matrice correspondant à un lien existant.

Il faut remarquer que cette matrice de similarité peut être vue comme un graphe, permettant d'appliquer des algorithmes de graphes pour la tache d'identification d'ensembles de pages apparentées. Toutefois, le graphe correspondant à la matrice de similarité peut être complètement(ou presque) connexe. Afin de réduire le nombre de

parasites, on met un certain seuil puis on supprime les arêtes ayant des fréquences très petites.

- Création du graphe :

Un graphe est créé où chaque page est un nœud et chaque cellule non nulle de la matrice est un arc. Ensuite on applique des algorithmes fournissant des informations sur la connexité du graphe(par exemple l'algorithme en temps linéaire pour l'identification des composants connexes). Les informations sur les valuations des arcs sont ignorés dans cette étape pour des raisons d'efficacité.

En créant un graphe clairsemé, et en utilisant un algorithme efficace pour l'analyse de groupe, on peut identifier des clusters très significatifs plus rapidement que quand on utilise les méthodes traditionnelles de regroupement. Nous pouvons extraire deux genres de sous-graphes menant à deux variantes de l'algorithme *page gather*.

- PGclique trouve les cliques maximales du graphe. Une clique est un sous graphe dans lequel chaque couple de sommets est lié par un arc. Une clique maximale n'est jamais un sous ensemble d'un autre clique. Pour des raisons d'efficacité, la taille des cliques découvertes est bornée.
- PGcc trouve tous les composants connexes, sous graphes dans lesquels chaque paire de pages est reliée par un chemin(suite d'arcs) entre elles.

Dans la deuxième étape de l'algorithme, on avait mis un seuil à la matrice de similarité. Si ce seuil est élevé, le graphe sera clairsemé, nous trouverons peu de clusters, qui seront de petites tailles et de grande qualité.

Si le seuil est bas, nous trouverons des clusters très grands.

Il faut noter que PGclique et PGcc ne seront pas utilisés avec les mêmes seuils. Un graphe clairsemé contenant beaucoup de composants connexes peut être trop clairsemé pour contenir des cliques.

- choix de cluster :

L'étape précédente peut faire sortir plusieurs clusters, et donc l'idéal serait d'en utiliser que quelques-uns uns. Les clusters trouvés sont évalués et triés par rapport aux moyennes des fréquences de cooccurrence entre les paires de pages dans les clusters. On trouve que PGclique a tendance à découvrir plusieurs clusters similaires. Donc deux techniques sont utilisées pour réduire le nombre de clusters similaires dans le résultat final :

- la réduction des chevauchements :qui consiste à se déplacer sur la liste rangée de cluster en supprimant ceux qui empiètent le plus sur un cluster déjà rencontré.
- la fusion :qui consiste à parcourir la liste rangée de cluster et à fusionner les paires de cluster qui se chevauchent suffisamment.

Toutes ces deux approches requièrent une mesure sur le chevauchement. On utilise le rapport entre les tailles de l'intersection et de l'union de deux clusters. Il faut noter que puisque les composants connexes ne se chevauchent jamais, ni la réduction, sur la fusion n'auront un effet sur PGcc.

La réduction et la fusion ont tous deux des avantages et des inconvénients. En général la réduction préserve la cohérence des clusters trouvés(puisqu'elle ne change pas leurs contenus) mais peut rater des pages qui pouvaient être mises dans le même cluster. La fusion, de son coté peut regrouper toutes les pages apparentées, mais au coût de réduire

la cohérence des clusters. Par défaut, la réduction est utilisée pour préserver la cohérence des clusters.

- Création des pages web :

L'algorithme *page gather* trouve les ensembles de liens candidats, et les présente au webmaster. L'administrateur est incité à accepter ou rejeter le cluster, à le nommer, à effacer tout lien qui lui semble inapproprié.

Les liens sont étiquetés avec les titres des pages cibles et ordonnés alphabétiquement.

Ceci n'est qu'une vision globale de l'algorithme, on pourrait avoir plusieurs variantes puisque les spécificités des sites peuvent varier d'un site à l'autre.

IV.3. Méthodologie CBR : raisonnement à partir de cas.

« to solve a new problem by remembering a previous similar situation and by reusing information and knowledge of that situation » ***[6].***

Le Case-Based Reasoning ou raisonnement à partir de cas est une méthodologie de résolution de problèmes fondée sur la réutilisation des expériences passées, appelées *Cas* pour la résolution de nouveaux problèmes. Un cas est composé de deux parties : la partie *problème* et la partie *solution*. La partie problème est composée d'un ensemble d'indices qui déterminent dans quelle situation un cas est applicable et utile. Les problèmes résolus sont stockés dans une base d'expériences dite *base de cas*. Lorsqu'un nouveau problème se présente ce problème est décrit par un cas dit *cas cible* où seule la partie problème est connue. La méthodologie du RàPC opère alors selon quatre phases séquentielles :

IV.3.1. La phase de remémoration(*case retrieval)* :

L'objectif est d'extraire de la base de cas des anciens cas dont la partie problème est similaire au problème à résoudre. Des mesures de similarités sont alors à définir sur les indices constituant la partie problème d'un cas. Les cas extraits de la base sont appelés les *cas sources*. Les cas sources sont alors passés à la phase d'adaptation.

IV.3.2. La phase de réutilisation(*case reuse*) :

L'objectif de cette phase est de proposer une solution au problème courant(le cas cible) en adaptant les solutions proposées par les cas sources. L'adaptation repose souvent sur l'utilisation de connaissances dans le domaine de l'application. A l'issue de cette phase, une ou plusieurs solutions seront proposées pour le cas cible.

IV.3.3. La phase de révision(*case revision*) :

L'objectif de cette phase est de réviser les solutions proposées par la phase précédente en fonction de certaines règles et /ou heuristiques, qui dépendent du domaine de l'application. La phase de révision peut être faite par des experts dans le domaine de l'application ou d'une manière automatique.

IV.3.4. La phase d'apprentissage(*case retainment-learning*) :

Cette phase a la charge d'enrichir l'expérience du sytème RàPC en enrichissant la base de cas par les nouveaux problèmes résolus(cas cible auquel on a apporté une solution). En effet les cas résolus peuvent être ajoutés à la base de cas et être utilisés dans des cycles de

raisonnement futurs. Cependant, avant d'ajouter ces cas, il faut juger la pertinence de cet ajout. Il faut éviter par exemple d'ajouter des cas redondants ce qui affecter les performances du système en termes de temps et de traitement sans pour autant améliorer la qualité des solutions apportées.

Une caractéristique très importante du CBR est son coté apprentissage. La notion de raisonnement à partir de cas ne dénote pas seulement une méthode particulière de raisonnement, elle dénote aussi une « machine learning » qui permet un apprentissage par mise à jour de la base de cas après qu'un certain problème a été résolu.

Quand un problème est bien résolu, l'expérience est retenue afin de résoudre des problèmes similaires dans le futur. Quand une tentative de résolution d'un problème échoue, les causes de léchec sont identifiées et mémorisées dans le but d'éviter la même erreur dans le futur.

Le raisonnement à partir de cas favorise l'apprentissage à partir d'expériences. Toutefois, l'apprentissage efficace avec cette méthodologie nécessite un ensemble de « bonnes » stratégies afin d'extraire des connaissances pertinentes à partir des expériences, d'intégrer un cas dans une structure de connaissance existante, et d'indexer le cas pour l'égaler à d'autres cas similaires.

IV.4. Adaptation structurelle des sites web : COBRA.

Un site web est dit site adaptatif s'il peut évoluer automatiquement en fonction de l'intérêt d'un ou plusieurs acteurs concernés par le site.

Dans **[5]** est décrite une approche d'auto adaptation des sites web. L'approche est appelée COBRA, une abréviation de **C**br-based c**O**llaborative **B**rowsing **A**ssistant.

Cobra est une approche d'auto adaptation qui applique à la fois une adaptation structurelle et une adaptation de la présentation des sites. En effet l'approche permet de guider les utilisateurs à naviguer dans un site en annotant les liens existant par rapport à leur intérêt pour l'utilisateur courant et en ajoutant des liens de raccourcis d'une manière adaptative.

L'approche est *orientée utilisateur*. En effet, l'approche Cobra consiste à observer le comportement de la navigation d'un utilisateur dans le site. Elle essaie de prédire les prochains mouvements de cet utilisateur en réutilisant des épisodes de navigations passées qui sont similaires au comportement de navigation de l'utilisateur actuel. La méthodologie RàPC est utilisée à cet effet. A l'issue du cycle de raisonnement à partir de cas, l'approche prédit un ensemble de pages susceptibles d'intéresser l'utilisateur. Certaines de ces pages sont directement liées par les liens hypertextes à la page courante ; dans ce cas ces liens seront mis en relief. D'autres pages prédites peuvent ne pas être connectées à la page courante. Dans ce cas des liens dynamiques seront ajoutés à la page courante. La prédiction des pages étant fondée sur l'analyse de similarité entre la navigation courante et les navigations passées des autres utilisateurs, l'approche applique alors une stratégie d'*apprentissage à partir de groupe* d'utilisateur. L'adaptation est faite en *ligne* et d'une manière personnelle.

Les navigations des utilisateurs dans le site seront enregistrées et stockées dans une base dite *base de navigations*. Un cas source décrit une expérience de navigation dans une navigation donnée. Avant de décrire le cycle de raisonnement appliqué par l'approche Cobra il convient de décrire d'abord la structure de sauvegarde de navigations utilisateurs et la structure des cas.

IV.4.1. Structure d'une navigation

Une navigation N est décrite par une structure qui comporte les champs suivants :

- **N**id : un identificateur qui identifie chaque navigation d'une manière unique.
- ADMIN : un champ qui porte des informations relatives à l'administration de la navigation comme par exemple la date de l'enregistrement de la navigation, la longueur de la navigation, l'adresse IP du client, la date de la dernière utilisation de la navigation par le mécanisme du raisonnement(voir ci-après), etc.
- LIST : est une liste chaînée qui décrit la séquence des pages visitées pendant la navigation. Cette liste est composée d'une séquence de nœuds. Chaque nœud représente une page visitée. Un nœud porte les informations suivantes : 1) l'adresse(l'URL) de la page et 2) une description du contenu de la page. Différentes manières peuvent être utilisées pour décrire le contenu d'une page. A titre d'exemple, le contenu d'une page peut être décrite par un vecteur de mots clés. Deux nœuds(pages) successifs sont connectés par un ou plusieurs *arcs*. Chaque arc correspond à une *hypothèse d'action qui peut justifier* la transition entre les deux pages. Quatre types d'action (ou hypothèse d'actions) sont définis :

1. Suivre le lien de rang i.
2. Suivre le lien dont l'ancre est A
3. Revenir *n* pas dans la navigation courante.
4. Aller vers l'URL U

La construction d'une navigation est faite comme suit : l'approche Cobra observe la succession des pages (ou URL) demandées par un utilisateur. Chaque page demandée est représentée par un nœud. Pour chaque page visitée, l'ensemble des liens hypertextes qu'elle contient est extrait. Ces liens sont stockés temporairement dans un *tableau de liens* qui donne pour chaque lien les informations suivantes : 1) l'ancre du lien et 2) l'URL destination de ce lien. Les liens sont stockés dans le tableau dans l'ordre de leur extraction de la page. Maintenant, lorsque l'utilisateur change de page, l'URL de la page demandée et le contenu du tableau de liens de la page précédente sont comparés. Les deux nœuds qui représentent les deux pages successives P1 et P2 seront reliées par :

- Des actions de types «*suivre le lien du rang i*» si l'URL du rang i dans le tableau de liens de P1 est égale à l'URL de la page P2. Il est à remarquer que plusieurs actions de ce même type peuvent lier deux pages. Afin de rendre l'approche plus souple face aux modifications des pages du site le rang de liens est décrit d'une manière relative au nombre total de liens dans la page. Ainsi si l'URL demandé correspond à l'URL de lien de rang i et au dernier lien de la page courante ;deux actions seront déduites : « *suivre le lien i/n* » et *suivre le lien n/n* » où n est le nombre total de liens contenus dans la page P1.
- Des actions de type «*suivre le lien dont l'ancre est A*» si l'URL de la page P2 est égal à un URL contenu dans le tableau de liens de la page P1 et dont

l'ancre est A. Encore une fois plusieurs actions de ce type peuvent connecter deux nœuds successifs.

- Des actions de type *« revenir n pas »* si l'URL de la page P2 est la même que l'URL d'une page Pn visitée n pas plus tôt dans la même navigation.
- Une action de type *« aller vers l'URL u»* où *u* est l'URL de la page P2. C'est l'action par défaut.

A chaque arc (action) est attribué un *facteur de confiance* qui mesure le degré de confiance du système dans la capacité à expliquer la transition de P1 à P2 . A la construction d'une navigation, à chaque action est associé un degré de confiance égal à 1/NA où NA est le nombre d'actions déduites pour relier les deux nœuds P1 et P2.

IV.4.2. Structure d'un cas

Un cas référence une expérience ou comportement dans une navigation donnée. Une structure composée des informations suivantes est proposée :

- **C**ID : c'est l'identificateur **N**ID de la navigation à partir de laquelle le cas est extrait.
- Index : c'est le rang de la page dans la navigation NID à laquelle le cas réfère. Autrement dit c'est l'instant dans la navigation qui détermine l'expérience de la navigation capturée par le cas.
- Historique : c'est une séquence d'au plus p pages (nœuds) qui précèdent immédiatement la page du rang INDEX dans la navigation NID.
- ACTIONS : c'est l'ensemble des actions qui relient la page du rang INDEX à la page qui la succède dans la navigation NID. Chaque action est associée à son degré de confiance.

Le couple INDEX, HISTORIQUE constitue la partie problème du cas. Le champ ACTIONS représente la partie solution.

IV.4.3. Phases du raisonnement

Une première phase consiste à *élaborer* le cas à partir d'une navigation courante, Nc, dans le site. L'élaboration de cas cible est faite comme suite :

- Le CID a la valeur de l'identificateur de la navigation Nc.
- L'index est le rang de la dernière page visitée dans la navigation Nc.
- L'HISTORIQUE est composé de la séquence de p pages (nœuds) visitées avant la page INDEX.
- La partie solution est vide.

Une fois le cas cible élaboré, le cycle de raisonnement peut commencer en appliquant les quatre phases du cycle RàPC :

Phase de remémoration.

Une opération de recherche est lancée dans la base de cas pour retrouver des cas dont le champ HISTORIQUE est similaire à l'historique du cas cible. La similarité est mesurée par une fonction d'agrégation de similarité entre les pages (nœuds) qui constituent les deux historiques. L'ordre des pages est pris en compte. La similarité entre deux pages est elle-même une agrégation entre la similarité sur le contenu des pages et la similarité entre les URL des pages. A l'issue de cette phase le cas dont la similarité avec le cas cible dépasse un certain seuil Sr est retenu. Les K cas les plus similaires sont passés à la phase de la réutilistation.

Phase de réutilisation.

Chaque cas source, Csi, retenu par la phase précédente propose un ensemble d'actions A(Csi). L'approche Cobra utilise ces ensembles d'actions de la manière suivante pour prédire le prochain mouvement de l'utilisateur : chacune des actions proposées est évaluée dans le contexte de la navigation courante Nc. L'évaluation d'une action permet de désigner une URL. Par exemple étant donné une action a de type *« suivre le lien n/n »*, l'évaluation de cette action dans le contexte de la navigation Nc donne l'URL destination du dernier lien hypertexte contenu dans la page courante dans la navigation Nc. Remarquer que l'évaluation des actions différentes peut donner la même URL en résultat. A chaque URL ainsi calculé est associé un facteur de confiance qui est fonction du nombre des actions dont l'évaluation a donné comme résultat cet URL, le degré de similarité entre le cas cible, le degré de similarité entre le cas cible et le cas source qui propose l'action et le degré de confiance dans l'action. Les différents URL obtenus après évaluation de toutes les actions proposées par les cas sélectionnés par la phase de remémoration seront triés en fonction de la valeur de leur facteur de confiance. Les URL dont le facteur de confiance dépasse un certain seuil F sont retenus. Ces URL constituent l'ensemble des URL prédits par le système. Si un URL prédit désigne une page liée par un lien hypertexte à la page courante dans Nc, alors ce lien va être mis en relief, sinon, l'URL est ajouté dans une section spéciale de liens recommandés.

Phase de révision.

Après avoir calculé les URL prédits, l'approche observe si l'utilisateur suit l'une des prédictions ou non. Dans le cas où une prédiction aurait été confirmée par l'utilisateur le degré de confiance dans les actions proposées par les cas sources retenus par la phase de remémoration sera augmenté. Les confiances dans les actions qui ont donné de fausses prédictions seront diminuées

Phase de l'apprentissage.

A la fin de chaque navigation l'approche ajoute la navigation courante à la base de navigation rendant ainsi possible de réutiliser les expériences de navigation contenues dans Nc pour des problèmes futurs. Des mesures d'évaluation de l'intérêt de l'ajout d'une navigation sont utilisées par l'approche mais elles ne sont pas présentées ici.

IV.5. Autour des motifs séquentiels:

La problématique de la recherche de motifs séquentiels consiste à détecter des comportements typiques dans le temps. Dans un tel contexte, un motif séquentiel peut être par exemple «10% des clients qui visitent les pages A et B, le même jour, visitent la page C moins de 2 heures après ».

Initialement introduite par **R. Agrawal** et **R. Srikant**, cette problématique est traitée par l'approche *GSP* (Generalized Sequential Patterns).

Considérons un ensemble de séquences appelées séquences de données, chaque séquence est une liste de transactions composées d'une estampille et d'un ensemble d'items. Il s'agit alors de rechercher toutes les séquences dont le nombre d'occurrences dans la base est supérieur à un support spécifié par l'utilisateur. Les séquences vérifiant cette contrainte sont appelées des séquences fréquentes ou motifs séquentiels. Dans **[10]**, la problématique

est étendue pour prendre en compte des contraintes temporelles.

Les séquences fréquentes vérifiant, en plus du support, ces contraintes sont appelées *motifs séquentiels généralisés*.

Dans **[9]** est proposé un nouvel algorithme, *PSP* (Prefix-tree for Sequential Patterns), qui reprend les principes fondamentaux de *GSP* mais utilise une structure de données différente pour optimiser la recherche des motifs. Au cours de différentes expérimentations, il a été montré dans **[8]** que PSP est plus performant que GSP.

IV.5.1. Définition préliminaires :

Dans **[8]**, la notion de séquence est définie de la manière suivante :

Définition : Une *transaction* constitue, pour un utilisateur U, l'ensemble des pages(items) visitées par U à une même date, elle est décrite sous forme d'un triplet : <ip-user, date-user, itemset>. Un *itemset* est un ensemble non vide d'items noté ($i1$ $i2$ …ik). Une *séquence* ou *motif séquentiel* est une liste ordonnée, non vide, d'itemsets notée <$s1$ $s2$….sn> où sj , 1<j<n, est un itemset (une séquence est donc une suite de transactions avec une relation d'ordre entre les transactions), représentant l'ensemble des visites d'un utilisateur. Une séquence regroupant k items est dite une *k-séquence*. Soit T1, T 2, …Tm les transactions d'un utilisateur, ordonnées par date de visite croissante et soit itemset(Ti), 1<i<m, l'ensemble des items correspondants à Ti , alors la séquence de données de cet utilisateur est <itemset(T1) itemset(T2)...itemset(Tm)>.

Exemple : Soit U un utilisateur et S=<(P1) (P2P3) (P4)>, la séquence de données représentant les visites de cet utilisateur. S peut être interprété par «U a visité la pageP1, puis, de façon simultanée, les pages P2 et P3 et enfin la page P4 ».

Définition : Soit s1=<$a1a2...an$> et s2=<$b1b$ $2...bm$> deux séquences de données. s1 est incluse dans $s2$ (s1<$s2$) si et seulement si il existe $i1$<$i2$<...<in des entiers tels que $a1 \subseteq bi1$, $a2 \subseteq bi2$, ... $an \subseteq bi$n.

Exemple : La séquence s1=<(3) (4 5) (8)> est incluse dans la séquence s1 =<(7) (3 8)

(9) (4 5 6) (8)> (i.e. s1< s2) car (3) $\subseteq$ (3 8), (4 5) $\subseteq$ (4 5 6) et (8) $\subseteq$ (8). En revanche <(3)(5)> $\not<$ <(3 5)> (et vice versa).

De manière générale, l'approche retenue par la plupart des algorithmes pour rechercher des

itemsets fréquents ou des motifs séquentiels consiste à chercher l'ensemble L1 d'items vérifiant le support de la base, puis à générer, à partir de celui-ci, un ensemble C2 de 2-items appelé candidats. Un nouveau parcours de la base permet de ne conserver que les candidats dont le support est supérieur au support minimum formant ainsi l'ensemble L2 des fréquents de taille 2. A partir de L2, de nouveaux candidats sont générés constituant l'ensemble C3 des candidats de taille 3. Le support associé à ces candidats est calculé par un autre examen de la base. Le principe énoncé s'applique tant qu'il est possible de trouver des fréquents parmi les candidats générés.

Telle qu'elle a été introduite, la notion de séquence présente une rigidité rédhibitoire pour certains types d'application. En effet, si l'intervalle de temps entre deux transactions est suffisamment court, il peut être alors envisageable de traiter celles-ci comme une unique transaction. De façon similaire, un intervalle trop long peut rendre inintéressant l'étude de la séquence des deux transactions. C'est pourquoi la notion de *séquences généralisées* a été introduite dans **[10]** pour pallier ces restrictions en proposant l'introduction de contraintes temporelles comme l'indique la définition suivante :

Définition : Soient *minGap* une durée minimum, *maxGap* une durée maximum et *windowSize* une durée de rectification, spécifiées par l'utilisateur. Une séquence de données $d = <d1...dm>$ supporte une séquence $s = <s1...sn>$ s'il existe des entiers $l1 \leq u1 < l2 \leq u2 < ... < ln \leq un$ tel que :

1. $si \subset \cup^{ui}_{k=li}\ dk$, $1 \leq i \leq n$;
2. date(d_{ui}) - date(d_{li}) $\leq$ *windowSize*, $1 < i < n$;
3. date(d_{li}) - date(d_{ui-1}) > *minGap*, $2 < i < n$;
4. date(d_{ui}) - date(d_{li-1}) $\leq$ *maxGap*, $2 < i < n$.

Le support de *s*, noté *supp(s)*, est le pourcentage de toutes les séquences dans *D* qui supporte *s*. Si *supp(s)* ³ *minsupp*, avec une valeur de support minimum *minsupp*, la séquence *s* est dite *fréquente.*

Ce concept de motifs séquentiels généralisés permet une manipulation plus souple des transactions des clients dans la mesure où il est désormais possible de :

- regrouper des itemsets lorsque leurs dates sont assez proches via la contrainte de *windowSize*,
- considérer des itemsets comme trop rapprochés pour apparaître dans la même séquence fréquente avec la contrainte de *minGap*,
- considérer des itemsets comme trop éloignés pour apparaître dans la même séquence fréquente avec la contrainte de *maxGap*.

Exemple : Considérons l'exemple suivant qui illustre la prise en compte des contraintes de temps sur une base de données D réduite à quatre clients (Cf. Figure suivante).

Avec un support minimum strictement supérieur à 50%, i.e. pour qu'une séquence soit supportée, elle doit être vérifiée par au moins deux séquences de la base de données, les motifs séquentiels sont, d'après la définition :<(20) (90)>,<(30) (90)> et <(30) (40; 70)>, les deux premières car elles peuvent être retrouvées pour les deux utilisateurs U1 et U4, et la dernière car elle intervient pour les utilisateurs U2 et U4.

Utilisateur	**Date**	**Items**	**Utilisateur**	**Date**	**Items**
U1	01/01/2001	20,60	**U3**	05/01/2001	30,50,70
U1	02/02/2001	20	**U3**	12/02/2001	10,20
U1	04/02/2001	30			
U1	18/02/2001	80,90			
U2	11/01/2001	10	**U4**	06/02/2001	20,30
U2	12/01/2001	30	**U4**	07/02/2001	40,70
U2	29/01/2001	40,60,70	**U4**	08/02/2001	90

Fig. Exemple de base de données

Si nous considérons une fenêtre de temps de 2 jours (*windowSize* = 2 jours), nous pouvons regrouper ensemble tous les items qui ont été visités dans un intervalle de 2 jours même s'ils sont dans des transactions différentes. Dans ce cas, une nouvelle séquence fréquente <(20 30) (90)> est extraite car elle se retrouve dans la première transaction de U4 tout en étant détectée pour U1 (avec un couple de transactions respectant le *windowSize*).

Considérons maintenant, du point de vue de l'utilisateur, que deux itemsets extraits successivement ne sont pas intéressant lorsqu'ils sont séparés par un intervalle de temps de 15 jours ou plus. Cette contrainte de *maxGap* a pour effet d'éliminer <(30) (40,70)> de l'ensemble des séquences fréquentes parce que, pour l'utilisation U2, il y a 17 jours entre les deux transactions.

La séquence de données de U2 ne satisfait pas la contrainte de *maxGap* et la contrainte de support minimal n'est donc plus vérifiée.

Considérons maintenant la séquence extraite via la contrainte de *windowSize*, i.e. <(20 30) (90)>, son premier itemset est composé de deux items différents intervenus dans un intervalle de temps de 2 jours. Si nous considérons l'item (30), nous voyons qu'il existe un intervalle de temps de 14 jours avec (90). Cependant, le délai observé est de 16 jours entre les items (20) et (90) donc la contrainte de *maxGap* n'est pas vérifiée. La séquence est alors supprimée de l'ensemble des séquences fréquentes.

IV.5.2. L'algorithme GSP – Generalized Sequential Patterns :

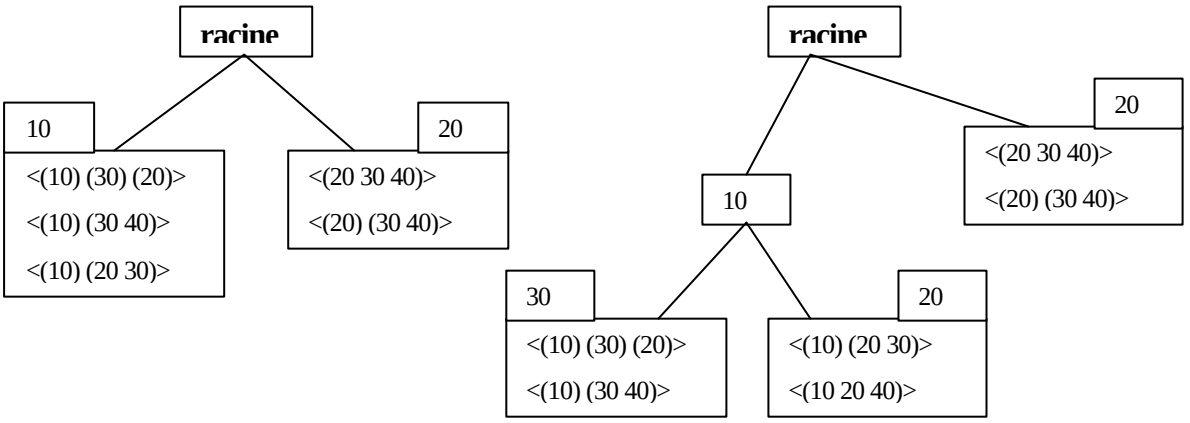
Pour construire les séquences candidates et les séquences fréquentes, l'algorithme *GSP* **[10]** procède selon différentes étapes comme la plupart des algorithmes de recherche des itemsets fréquents. Il s'agit, tout d'abord, de calculer le support de chaque item de la base. A la fin du parcours de celle-ci, les items fréquents, i.e. vérifiant la contrainte de support minimum, sont découverts. Ils sont considérés comme des 1-séquences fréquentes (séquence composée d'un itemset réduit à un singleton). Cet ensemble initial est le point de départ de la deuxième étape. L'ensemble des 2-séquences candidates est alors construit en combinant tous les couples d'items fréquents, à la fois dans la même transaction et dans une transaction séparée.

A partir de ce point, toute étape k (k > 2) réalise, à partir d'un ensemble de (k-1)-séquences fréquentes, les deux sous étapes suivantes :

- la première sous étape(*join phase*) concerne la génération des candidats. L'idée principale est de retrouver parmi toutes les séquences dans L_{k-1}, des couples de séquences (*s*,*s'*) tels qu'en éliminant le premier item de *s* et le dernier item de *s'*, nous retrouvions les deux mêmes sous-séquences. Quand un couple de séquence (*s*, *s'*) vérifie la condition, un nouveau candidat est construit en ajoutant le dernier item de *s'* à *s*. Dans la séquence candidate, ajoutée à C_k, le découpage des transactions est respecté.
- la seconde sous-étape (*prune phase*) a pour but d'obtenir l'ensemble L_k des k-séquences fréquentes en éliminant de C_k les séquences ne vérifiant pas le support minimum. Il s'agit donc de compter le nombre de séquences candidates intervenant dans les séquences de données.

Les séquences candidates sont stockées dans une structure de *hash-tree* afin d'assurer un accès efficace. Plus précisément, les séquences candidates sont stockées dans les feuilles de l'arbre alors que les nœuds intermédiaires contiennent des tables de hachage pour optimiser la recherche. Chaque séquence de données *d* de la base de données est hachée pour trouver les candidats contenus dans *d*. En parcourant une séquence de données, les contraintes de temps doivent être également prises en compte. Ceci est réalisé en parcourant l'arbre de « haut en bas » et de « bas en haut » afin d'appliquer une première fois les contraintes de temps. Le parcours de l'arbre permet d'obtenir des ensembles de feuilles contenant des candidats potentiels. A partir de ces feuilles, *GSP* recherche tous les candidats effectivement présents dans la séquence de données et qui vérifient les contraintes de temps. Pour cela, chaque candidat est stocké dans une nouvelle structure et est comparé avec la séquence de données. La comparaison a lieu en parcourant en avant et en arrière la séquence candidate et la séquence de données. La phase « avant » est réalisée en examinant les items progressivement. Pendant cette phase, la condition de *minGap* est appliquée pour éliminer les itemsets qui sont trop proches de leur précédent. Tout en parcourant les itemsets, *windowSize* est appliqué pour redéfinir les limites des transactions. La phase « retour arrière » est nécessaire lorsque la contrainte de *maxGap* n'est plus vérifiée. Dans ce cas, il est nécessaire d'éliminer tous les itemsets pour lesquels *maxGap* n'est pas vérifié et de recommencer un parcours avant à partir du dernier item qui vérifiait la condition de *maxGap*. Toutes les séquences fréquentes sont alors stockées dans une nouvelle structure de *hash-tree* multi-niveau pour permettre d'optimiser la génération des candidats.

Exemple : La figure suivante illustre la façon dont *GSP* gère la structure de *hash-tree*. Les deux arbres servent à conserver les mêmes candidats mais les feuilles de l'arbre du haut peuvent contenir plus de deux séquences candidates alors que dans l'arbre de droite seulement deux séquences candidates peuvent être stockées dans la même feuille. Nous remarquons que la feuille qui porte l'étiquette 20 dans l'arbre de gauche (le deuxième fils de la racine) est utilisée pour stocker les séquences candidates <(20 30 40)> et <(20) (30 40)>. Quand *GSP* atteint cette feuille, il n'a aucun moyen de savoir si c'est la séquence <(20 30 40)> ou bien la séquence <(20) (30 40)> qui l'a conduit jusqu'`a cette feuille. C'est pour cette raison que GSP doit tester les séquences candidates contenues dans les feuilles atteintes afin de savoir quel(s) support(s) incrémenter.



IV.5.3. L'algorithme PSP – Prefix-tree for Sequential Patterns :

Dans **[9]** est proposée l'approche *PSP* (Prefix-tree for Sequential Patterns) dont l'originalité est d'utiliser une structure différente de celle adoptée par *GSP*. C'est ainsi qu'une nouvelle organisation des séquences candidates permet de trouver plus efficacement l'ensemble des candidats inclus dans une séquence de données. En effet, la structure de *hash-tree* utilisée dans *GSP* consiste à mettre en commun des préfixes sans faire de distinction entre deux items du même itemset et deux items avec changement d'itemsets. De ce fait, il se voit contraint, lorsqu'il atteint une séquence, de tester si les dates des items qu'elle représente sont adéquates par rapport à la séquence de données testée.

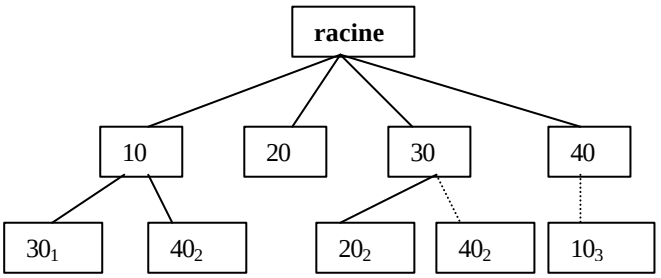
Par contre, la structure de données utilisée par *PSP* est une structure de *prefix-tree*. Elle factorise les séquences candidates selon leur préfixe et tient compte des éventuels changements d'itemsets. Elle offre deux avantages principaux :

1. les séquences fréquentes sont stockées dans le même arbre que les séquences candidates afin d'optimiser la génération des candidats ;
2. contrairement à *GSP* qui nécessite une phase supplémentaire pour gérer les contraintes de temps, celles-ci peuvent être prises en compte directement lors du parcours de l'arbre.

De manière générale, à la $k^{ième}$ étape, l'arbre est d'une profondeur k. Il stocke toutes les *k-séquences* candidates de la manière suivante : chaque branche, de la racine à une feuille correspond à une séquence candidate et en considérant une simple branche, chaque nœud à la profondeur l ($k \geq l$) correspond au $l^{ième}$ item de la séquence. En outre, le support de la séquence de la racine à une feuille est associé à chaque nœud terminal. Pour distinguer les itemsets à l'intérieur d'une séquence (par exemple, (10 20) et (10)(20)) les fils d'un nœud sont séparés en deux catégories : « *same-transaction* » ou «*other-transaction* » (branche en traits pleins dans la figure de l'exemple ci dessous).

Exemple : Considérons l'ensemble de 2-séquences fréquentes suivant : $L2$ = {<(10)(30)>,<(10) (40)>,<(30) (20)>, <(30 40)>,<(40 10)>}. Il est organisé dans la structure

Prefix-tree comme illustré ci dessous. Chaque nœud terminal contient un item et un compteur. Si nous considérons le nœud correspondant à l'item 40, la valeur 2 associée indique que deux occurrences de la séquence {<(10) (40)>} ont déjà été détectées. L'item 20, fils de la racine, est un fréquent de longueur 1.



Dans **[8]** sont décrites des expérimentations sur les algorithmes *GSP* et *PSP* en GNU C++.

Les observations montrent que *PSP* est plus performant que *GSP*, que ce soit avec des données synthétiques ou réelles.

V. Conclusion

Nous avons présenté dans ce document le *web data mining*. La liste des méthodologies et algorithmes relatés ici est loin d'être exhaustive ; mais la quasi-totalité des branches de recherche dans le domaine du *web data mining* a été décrite.

Le choix d'un algorithme ou d'une méthodologie pour l'analyse des utilisations d'un site dépend principalement des spécificités du site et des besoins du webdisigner.

Le centre d'informations sur les pêches en guinée se présente sous la forme de site web générant du contenu dynamique.

Plusieurs algorithmes(mais pas tous) décrits dans ce document auraient pu être utilisés pour le CI. Par exemple, l'analyse du comportement d'un utilisateur singulier ne sera pas faisable, puisque dans le contexte du CI, il serait difficile de voir un même internaute qui « traverse » à chaque fois les mêmes liens(une information visualisée est supposée acquise). Par contre le clustering sur le contenu ou sur les utilisateurs peut être très utile.

L'idéal serait donc de sélectionner les approches intéressantes et utilisables et de procéder à une série d 'expérimentations pour voir lesquelles choisir.

VI. Références bibliographiques

[1] J. Borges, M. Levene.
« Data Mining of User Navigation Patterns »
Department of Computer Science, University College London,

[2] T. W. Yan, M. Jacobson, H. Garcia-Molina, U. Dayal
« From User Access Patterns to Dynamic Hypertext Linking »
First International World Wide Web Conference
May 6-10, 1996, Paris, France

[3] J. Srivastava, R. Cooley, M. Deshpande, P. N. Tan.
« Web Usage Mining:Discovery and Applications of Usage Patterns from Web Data »
Department of Computer Science and Engineering,
University Of Minnesota, Jan 2000.

[4] M. Perkowitz, O. Etzioni.
« Towards Adaptative Web Sites : Conceptual Framework and Case Study »
Department of Computer Science and Engineering,
University of Washington, Seattle, Jan 1998.

[5] R. Kanawati*, M. Malek**.
« **C**OBRA: Une approche d'adaptation structurelle des sites Web

fondée sur une technique d'apprentissage à partir des traces d'accès utilisateur et utilisant la méthodologie du raisonnement à partir de cas »
**LIPN, CNRS UPSERA 7034, Université Paris Nord*
***EISTI-LAPI*

[6] A. Aamodt, E. Plaza.
« Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches »
Published in: AI Communication, Vol.7 Nr. 1, March 1994, pp 39-59

[7] R. Cooley, P. N. Tan, J. Srivastava.
« Discovery of Interesting Usage Patterns from Web Data »
Department of Computer Science and Engineering,
University of Minnesota

[8] F. Masseglia, P. Poncelet, R. Cicchetti.
« An efficient algorithm for Web usage mining »
Networking and Information System Journal.
Volume X – n° X/2000, pages 1 à X

[9] F. Masseglia, F Cathala, P. Poncelet.
« The PSP Approach for Mining Sequential Patterns »
Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98), LNAI, Vol 1510
Nantes, France, September 1998, p. 176-184.

[10] R. Srikant, R. Agrawal.
« Mining Sequential Patterns: Generalizations and Performance Improvements »,
Proceedings of the 5nd International Conference on Extended Database Technology (EDBT'96),
Avignon, France, Septembre 1996, p. 3-17

[11] L. Catledge, J. Pitkow.
« Characterizing browsing behaviour on the world wide web »,
Computer Networks and ISDN Systems, 27(6), 1995.