Classification dynamique par treillis de concepts pour la recherche d'information sur le web.

Emmanuel Nauer, Yannick Toussaint

Classification dynamique par treillis de concepts pour la recherche d'information sur le web.

Emmanuel Nauer et Yannick Toussaint

LORIA – UMR 7503. B.P. 239, F-54506 Vandœuvre-lès-Nancy cedex, France.
{Emmanuel.Nauer,Yannick.Toussaint}@loria.fr

RÉSUMÉ. *L'analyse de concepts formels (ACF) permet d'organiser des objets en fonction de leurs propriétés. Des travaux récents ont utilisé l'ACF pour réorganiser, sous la forme d'un treillis de concepts, les réponses fournies par un moteur de recherche du web. L'utilisateur navigue dans le treillis pour explorer un résultat structuré et synthétique. Or, un tel treillis contient des concepts qui sont pertinents par rapport à une tâche de recherche d'information donnée et d'autres qui ne le sont pas. Pour que l'utilisateur puisse se focaliser sur ce qui l'intéresse et éliminer ce qui ne l'intéresse pas, nous proposons un système interactif dans lequel il va exprimer son intérêt (positif ou négatif) pour certains concepts du treillis. Ce contrôle de pertinence est exploité dans la classification pour faire évoluer le treillis et ainsi mieux l'adapter au besoin de l'utilisateur.*

ABSTRACT. *This paper presents an iterative and interactive information retrieval system to search on the web using formal concept analysis (FCA). FCA provides a natural way to organise objects according to their properties and it has been used in recent works to organise answers provided by a search engine according a concept lattice. The navigation into the lattice helps the user to explore a structured and synthetic result. Such a lattice contains concepts that are relevant and some others that are not relevant for a given information retrieval task. We introduce lattices into an interactive and iterative system. The user expresses his negative or positive agreement with some concept of the lattice.These user choices modify the object set and the properties used as input of the classification process. The lattice evolves during the process in order to better fit user need.*

MOTS-CLÉS : *Classification, treillis de concepts, contrôle de pertinence utilisateur, web*

KEYWORDS: *Classfication, concept lattices, user relevance feedback, web*

1. Introduction

L'analyse de concepts formels (ACF) est une méthode de classification qui permet d'organiser hiérarchiquement les objets à partir de leurs propriétés. En recherche d'information (RI), l'ACF a notamment été appliquée à la reformulation de requêtes, à l'ordonnancement de réponses, à la classification de documents. Dans le cadre du web, des travaux récents (Carpineto *et al.,* 2004, Koester, 2006) ont proposé de construire un treillis de concepts à partir des mots du titre et de l'extrait des documents retournés par un moteur de recherche (GOOGLE) afin d'organiser ces documents dans une structure hiérarchique plus synthétique. Cette hiérarchie est cependant statique : l'ensemble des documents reste inchangé au cours d'une session de RI et il en est de même pour l'ensemble des propriétés. Or, la hiérarchie contient souvent des concepts (et des documents) non pertinents. De même, des documents pertinents, autres que ceux ayant servi à construire le treillis, peuvent exister sur le web.

Intégrer l'utilisateur dans le processus de RI en lui donnant la possibilité d'apprécier l'intérêt positif ou négatif des concepts et des documents est un moyen d'ajuster le résultat du système de recherche d'information (SRI) à son besoin. Le système CRECHAINDO, présenté dans ce papier, couple un contrôle de pertinence utilisateur à une navigation par treillis. L'utilisateur explore le treillis et identifie des concepts pertinents et/ou non pertinents. Ces retours utilisateur sont exploités pour modifier l'ensemble des documents à partir duquel le treillis est construit, et faire évoluer le treillis par la même occasion.

Ce papier est organisé de la façon suivante : la section 2 montre comment l'ACF et le retour utilisateur permet d'améliorer la RI, la section 3 détaille le système CRECHAINDO et la section 4 donne un exemple concret d'utilisation de CRECHAINDO. Les contributions significatives de CRECHAINDO et les perspectives de recherche de ce travail concluent le papier.

2. Améliorer la recherche d'information par l'analyse de concepts formels

2.1. *Analyse de concepts formels*

L'ACF est une approche mathématique pour l'analyse de données qui exploite la théorie des treillis. Un *contexte formel* est un triplet $\mathcal{K} = (G, M, I)$, où G est un ensemble d'objets, M un ensemble de propriétés et I la relation sur $G \times M$ exprimant qu'un objet possède une propriété (Ganter *et al.,* 1999). Le tableau présenté en Figure 1 donne un exemple de contexte : G est un ensemble de 6 documents ($\mathtt{d}_1$,...,$\mathtt{d}_6$) et M un ensemble de 7 propriétés qui sont des mots-clés décrivant les documents. Un *concept formel* est une paire (I, E), où E est l'ensemble maximal d'objets (appelé *extension*) possédant toutes les propriétés de I, et I est l'ensemble maximal de propriétés (appelé *intension*) partagées par tous les objets de E. Par exemple, $(\{\mathtt{IR}, \mathtt{MoteurDeRecherche}, \mathtt{Web}\}, \{\mathtt{d}_1, \mathtt{d}_2, \mathtt{d}_3\})$ est un concept formel (cf. diagramme présenté en Figure 2). De plus, l'ensemble $\mathcal{C}_\mathcal{K}$ de tous les concepts formels du contexte $\mathcal{K} = (G, M, I)$ est partiellement ordonné par l'inclusion des exten-

	Web	IR	MoteurDeRecherche	Annuaire	Yahoo!	Altavista	Google
d_1	×	×	×	×	×		
d_2	×	×	×				
d_3	×	×	×				
d_4	×			×			
d_5		×	×			×	×
d_6		×	×				

Figure 1. *Un contexte formel de 6 documents décrits par 7 propriétés.*

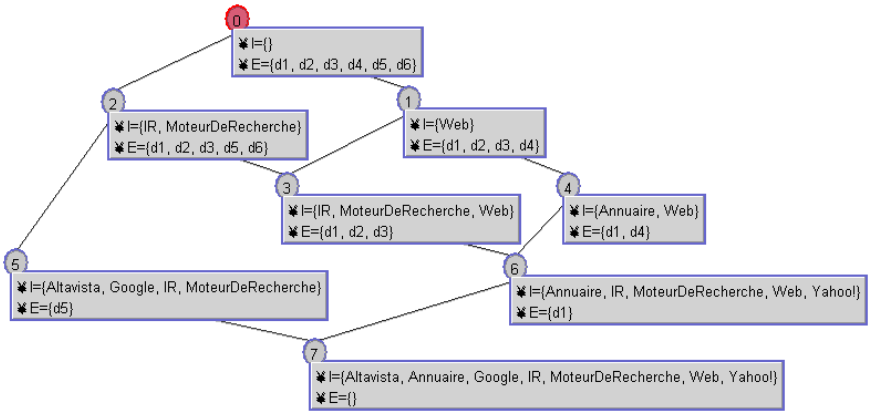

Figure 2. *Le treillis de concepts correspondant au contexte formel donné en Figure 1.*

sions. Cette relation dite de *spécialisation* entre concepts est notée $\leq_{\mathcal{K}}$. $\mathcal{L} = \langle \mathcal{C}_{\mathcal{K}}, \leq_{\mathcal{K}} \rangle$ est un treillis complet, appelé *treillis de concepts*. Le treillis $\mathcal{L}$ peut être représenté par un diagramme de Hasse dans lequel les nœuds sont les concepts et les arêtes, les liens de spécialisation/généralisation. La Figure 2 illustre le treillis de concepts correspondant au contexte formel donné en Figure 1. Le concept le plus général (dans la figure, tout en haut) contient tous les documents ; son intension est vide car aucune propriété n'est commune à tous les documents. De façon duale, le concept le plus spécifique est défini par l'ensemble de toutes les propriétés. Dans notre exemple, son extension est vide, car aucun document n'est décrit par toutes les propriétés. De nombreux algorithmes ont été proposés pour la construction de treillis de concepts (cf. (Ganter *et al.*, 1999)). Pour notre application, nous utilisons la plate-forme CORON[1], qui implémente une large collection d'algorithmes pour la fouille de données symboliques, incluant des algorithmes de construction de treillis de concepts (Szathmary *et al.*, 2005).

1. `http://coron.loria.fr/`

2.2. *L'analyse de concepts formels pour la recherche d'information*

(Carpineto *et al.*, 2004) présente un état de l'art concernant l'utilisation de l'ACF en RI. L'idée phare est que l'intension d'un concept est vue comme une requête et son extension contient les documents, réponses à la requête. De plus, les concepts voisins d'un concept "requête" peuvent être vus comme les changements minimaux pour une reformulation. Le système REFINER (Carpineto *et al.*, 1998) exploite ces propriétés pour ne construire que la partie du treillis autour d'un concept "requête" et la présenter à l'utilisateur.

Les treillis de concepts sont également de bons candidats pour des systèmes de recherche d'information hybrides exploitant conjointement un accès par requête et un accès par navigation (Cole *et al.*, 2003, Ducrou, 2007). La requête de l'utilisateur est classifié dans le treillis. Ce concept est le point d'entrée pour naviguer dans la structure hiérarchique. Dans ce contexte, (Carpineto *et al.*, 1996) propose également d'améliorer la structure du treillis, en exploitant une hiérarchie de spécialisation (thésaurus) sur l'ensemble des termes décrivant les documents.

Deux autres fonctionnalités pour l'interrogation et la navigation sont proposées dans (Ducrou *et al.*, 2006). La première est une fonction de requête par l'exemple. Elle retourne les documents qui possèdent les propriétés communes aux documents exemples. La seconde est une mesure de similarité entre concepts calculée à partir de leurs intensions et de leurs extensions. Cette mesure permet d'ordonner les concepts en fonction de leur similarité avec un concept donné.

Enfin, les treillis ont également été utilisés pour classer les documents retournés par un SRI ou encore réorganiser la liste de documents fournie en réponse. Ainsi, (Carpineto *et al.*, 2000) propose d'ordonner les documents-réponses en considérant que plus les concepts sont proches du concept représentant la requête, plus les documents appartenant à leur extension sont pertinents. De son coté, le système CREDO, acronyme de *Conceptual REorganisation of DOcuments* (Carpineto *et al.*, 2004) ou son adaptation CREDINO pour les PDA (Carpineto *et al.*, 2006) exploite les treillis pour organiser l'ensemble des documents retournés par GOOGLE à une requête donnée et permettre à l'utilisateur de naviguer dans cet ensemble. Chaque document fourni en réponse par GOOGLE, tel que celui donné en illustration en Figure 3, est composé d'un titre, d'un extrait et d'un URL. Les mots qui constituent le titre et l'extrait des documents font office de propriétés. Un problème de la construction de treillis à partir de données textuelles est la génération d'un trop grand nombre de concepts. Ce problème résulte d'une combinatoire élevée dans la cooccurence de mots dans les do-

Figure 3. *Un document* GOOGLE *retourné pour une requête sur* "carpineto romano".

cuments. De plus, certains concepts dépourvus de sens peuvent être générés en raison de mauvaises combinaisons de termes dans les documents. Une façon de résoudre ces problèmes est de décrire un document par un ensemble limité de mots. Ce choix de restriction a été fait dans CREDO pour construire le premier niveau de la hiérarchie en exploitant le contexte *document* $\times$ *mots du titre*, donnant par la même occasion une plus grande importance aux mots des titres par rapport aux mots des extraits. Un nouveau contexte *document* $\times$ *(mots du titre + de l'extrait)*, intégrant les mots des extraits, servent à la construction des autres niveaux de la hiérarchie. Un algorithme spécifique est proposé pour construire une hiérarchie à partir de deux contextes. Dans la hiérarchie proposée par CREDO, l'intension d'un concept est un ensemble de mots et l'extension associée est l'ensemble des documents qui contiennent tous les mots de l'intension. La liste initiale de documents retournée par GOOGLE est à présent organisée de façon structurée. Le treillis est présenté à l'utilisateur sous forme d'arbre qui sert de support à l'exploration de l'ensemble des documents. L'accès aux documents se fait en sélectionnant des concepts de plus en plus spécifiques, auxquels sont rattachés des documents. Un intérêt majeur du treillis est l'héritage multiple entre concepts ; un même concept, et donc un même ensemble de documents, est atteignable par plusieurs chemins (i.e. un concept avec héritage multiple se retrouve dupliqué dans l'arbre).

Une exploitation par ACF des documents retournés en réponse par un moteur de recherche du web, avec l'objectif d'améliorer la RI sur le web, est également mise en œuvre dans deux autres outils :

– FooCA (Koester, 2006) permet de mieux appréhender le contexte associé aux documents, le présentant sous la forme d'un tableau comme celui de la figure Figure 1. Le treillis obtenu à partir du contexte est également visualisable sous la forme d'un graphique. De nombreuses options permettent de contrôler la création de ce contexte (nombre de documents demandés au moteur de recherche, nombre minimum d'objets par propriétés, utilisation ou non d'une lemmatisation, suppression ou non de mots vides, etc.). L'utilisateur peut reformuler son besoin en ajoutant un mot de façon positive ou négative à sa requête initiale, et le processus repart à partir du nouvel ensemble de documents.

– SearchSleuth (Ducrou *et al.*, 2007) se focalise exclusivement sur la reformulation de requête. La classique interface de recherche — à un champ — des moteurs de recherche du web est étendue pour proposer des reformulations potentielles à l'utilisateur. À l'instar de REFINER, les concepts génériques les plus spécifiques et les concepts spécifiques les plus généraux servent à proposer des termes pour généraliser ou spécialiser la requête. La mesure de similarité entre concepts introduite dans (Ducrou *et al.*, 2006) sert à proposer d'autres reformulations potentielles.

Notre travail reprend l'aspect interactif proposé dans FooCA et SearchSleuth pour l'adapter à, et améliorer, CREDO. Une originalité de notre travail est d'intégrer un processus de RI itératif. Ainsi, contrairement aux approches de FooCa, SearchSleuth et CREDO, le contexte et le treillis ne sont pas réinitialisés mais affinés à chaque interaction de l'utilisateur.

2.3. *Un processus de recherche d'information itératif et interactif*

L'objectif d'un SRI est de fournir à un utilisateur de l'information relative à un certain besoin. Dans la plupart des SRI, et en particulier dans les moteurs de recherche du web, le processus de RI consiste à soumettre une requête représentant le besoin de l'utilisateur. Mais, comme une requête est une représentation réduite du besoin de l'utilisateur, certains SRI exploitent de l'information additionnelle. Le contrôle de pertinence (*relevance feedback*, en anglais) (Rocchio, 1971) est un moyen de fournir plus d'information sur la recherche pour améliorer la précision de la recherche (Salton *et al.*, 1990).

Il existe deux grandes catégories de contrôle de pertinence : le contrôle de pertinence *explicite* et le contrôle de pertinence *implicite*. Dans le cas d'un contrôle de pertinence explicite, l'utilisateur doit *explicitement* exprimer son contrôle au système, par exemple, en entrant des mots-clés complémentaires, en répondant à des questions spécifiques, en identifiant des sous-ensembles de documents pertinents et/ou non pertinents (Kassab *et al.*, 2005), en annotant des documents (Dmitriev *et al.*, 2006), etc. Dans le cas d'un contrôle de pertinence implicite, les SRI ne proposent pas d'interactions spécifiques pour le contrôle. Le contrôle de pertinence est déduit par le SRI à partir de toutes les interactions *implicites* de l'utilisateur (Shen *et al.*, 2005). Par exemple, dans un système utilisant une formulation du besoin par requête, les résultats retournés en reponse à la première requête peuvent ne pas être satisfaisants. Souvent, l'utilisateur va interagir avec le système pour modifier sa requête initiale ou visualiser certains documents dans la liste des réponses produite. Toutes ces interactions peuvent être exploitées pour le contrôle de pertinence. La reformulation de requête peut, par exemple, désambiguïser le contexte de mots polysémiques (Dumais *et al.*, 2004). Nous renvoyons à (Kelly *et al.*, 2003) pour une classification des techniques de contrôle de pertinence implicite et leur exploitation dans les travaux majeurs du domaine.

Le processus de RI proposé dans CRECHAINDO implémente un contrôle de pertinence explicite. L'utilisateur interagit itérativement avec le système pour évaluer si un concept du treillis est pertinent ou non, par rapport à son besoin. Son jugement sert à modifier le contexte utilisé pour construire le treillis.

3. Le système CRECHAINDO

3.1. *Principes*

Pour opérationnaliser l'itérativité et l'interactivité du processus de RI, CRECHAINDO [2] intègre une modification dynamique du contexte. À la première étape du processus de RI, l'utilisateur soumet une requête et un treillis est construit. Par la suite, l'utilisateur va émettre une appréciation sur certains concepts du treillis. À l'instar des SRI dans lesquels l'utilisateur peut accepter/rejeter des documents ou ensembles de

2. Prototype expérimental accessible à partir de : `http://intoweb.loria.fr/CreChainDo`

documents, CRECHAINDO propose à l'utilisateur de sélectionner les concepts qui sont pertinents et ceux qui ne le sont pas.

Définition 1 *Un concept C est pertinent, si l'utilisateur estime qu'une requête Q, formée de la conjonction de tous les mots composant l'intension de C, est suceptible de retourner de nouveaux documents pertinents.*

L'idée sous-jacente de cette définition est que le treillis ne contient pas forcément, à un instant donné, tous les documents pertinents qui pourraient être trouvés sur le web, et qu'une nouvelle interrogation d'un moteur de recherche avec la requête Q peut retourner de nouveaux documents pertinents. En effet, la requête initiale n'est souvent pas parfaite ; l'utilisateur doit la reformuler et/ou la compléter pour préciser son besoin. La navigation dans le treillis guide par ailleurs l'utilisateur pour cette reformulation.

Définition 2 *Un concept est non pertinent si l'utilisateur estime que tous les documents contenus dans son extension ne sont pas pertinents.*

CRECHAINDO propose à l'utilisateur de sélectionner les concepts pertinents et non pertinents. Chacun de ses choix modifie le contexte et un nouveau treillis est calculé. Dans CRECHAINDO, le processus de RI est ainsi une succession de contexte : $K_0 \rightarrow K_1 \rightarrow \ldots \rightarrow K_i \rightarrow K_{i+1} \rightarrow \ldots$, avec $K_i = (G_i, M_i, I_i)$, le contexte de l'étape i. $K_0 = (G_0, M_0, I_0)$ est le contexte initial vide : $G_0 = \emptyset$, et par conséquent $M_0 = \emptyset$ et $I_0 = \emptyset$.

Nous détaillons maintenant comment le contexte peut évoluer.

3.2. *Extension de contexte*

L'objectif d'une extension de contexte est d'améliorer la précision de la RI, par la recherche de nouveaux documents, ainsi que la structuration de la hiérarchie par la prise en compte de ces nouveaux documents dans la construction du treillis.

Soit Q une requête utilisateur. Q est un ensemble non vide de mots : $Q = \{mot_i | i > 0\}$. Soit $DOC(Q)$ l'ensemble des documents retournés par un moteur de recherche pour la requête Q : $DOC(Q) = \{doc_j | j \geq 0\}$.

Définition 3 *L'extension du contexte K_i au contexte K_{i+1}, noté $K_i \xrightarrow{+Q} K_{i+1}$, est réalisée en ajoutant les réponses retournées par un moteur de recherche pour la requête Q :*

– les nouveaux documents sont ajoutés à l'ensemble d'objets du contexte : $G_{i+1} = G_i \cup DOC(Q)$;

– les nouveaux mots sont ajoutés à l'ensemble des propriétés du contexte : $M_{i+1} = M_i \cup \{mot_i | i \in DOC(Q)\}$;

– la relation I_i entre G_i et M_i est étendue à la relation I_{i+1} d'appartenance des mots (de M_{i+1}) aux documents (de G_{i+1}).

Dans un contexte K_{i+1} qui vient d'être *étendu*, les objets de G_i sont décrits exactement avec les mêmes propriétés. C'est pourquoi, pour tous les $(g, m) \in I_i$, si $g \in G_{i+1}$ alors $(g, m) \in I_{i+1}$. Pour tous les concepts $(g, m_i) \in K_i$, il existe un concept $(g, m_{i+1}) \in K_{i+1}$ avec $m_i \subseteq m_{i+1}$. L'intension reste inchangée tandis que l'extension peut augmenter. Ceci assure une évolution progressive du treillis entre deux étapes.

Il y a deux cas dans lesquels le contexte est étendu :

– l'utilisateur soumet une nouvelle requête : la requête Q_0 ($K_0 \xrightarrow{Q_0} K_1$) défini un nouveau contexte, mais l'utilisateur peut également introduire à n'importe quel moment du processus de RI une nouvelle requête Q_i entre deux étapes ($K_i \xrightarrow{+Q_i} K_{i+1}$), ajoutant de nouveaux mots dans la formulation de sa recherche ;

– l'utilisateur évalue un concept C comme étant pertinent pour sa recherche : une requête composée de la conjonction de tous les mots de l'intension de C est soumise à un moteur de recherche, le contexte est étendu par $K_i \xrightarrow{+intension(C)} K_{i+1}$.

3.3. *Réduction de contexte*

L'objectif d'une réduction de contexte est d'éliminer dans le treillis, les informations — concepts et documents — non pertinentes pour l'utilisateur. Une réduction est appliquée à chaque fois que l'utilisateur évalue un concept comme étant non pertinent. Les documents rattachés à ce concept, qui par définition ne sont plus pertinents, ne sont plus pris en compte dans la construction de la hiérarchie ; ce qui est un moyen de la *nettoyer*.

Définition 4 *La réduction du contexte K_i au contexte K_{i+1}, notée $K_i \xrightarrow{-C} K_{i+1}$, est réalisée comme suit :*

– les documents appartenant à l'extension de C sont supprimés du contexte : $G_{i+1} = G_i \setminus extension(C)$;

– la relation I_{i+1} entre G_{i+1} et M_{i+1} est obtenue en supprimant de I_i les lignes associées aux documents appartenant à $extension(C)$.

3.4. *Implémentation de* CreChainDo

L'interface utilisateur (cf. Figure 4) est divisée en 4 parties :

– tout en haut, l'interface d'interrogation permet à l'utilisateur de ① soumettre une requête et de spécifier quelques paramètres tels que ② le nombre de documents sou-

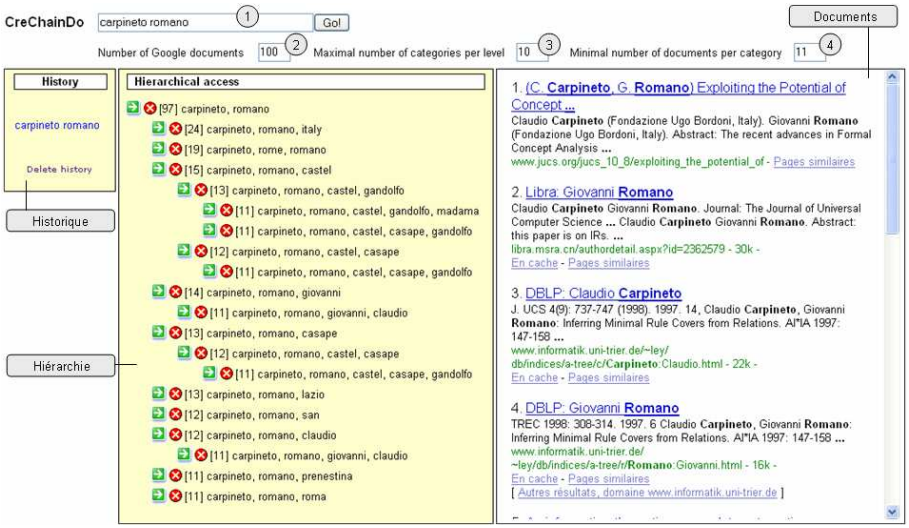

Figure 4. *L'interface de* CRECHAINDO*, après une requête sur "carpineto romano".*

haités en réponse de l'interrogation GOOGLE. Deux paramètres associés à la construction de la hiérarchie et de son affichage sont disponibles : ③ le nombre maximal de concepts (`Mnc`) affichés à chaque niveau de la hiérarchie et ④ le nombre minimal de documents (`mnd`) contenus dans l'extension des concepts (cette valeur peut être exprimée par un nombre ou en pourcentage du nombre de documents de contexte).

– au mileu de l'interface, la hiérarchie résultant d'un parcours en profondeur du treillis présente, sous forme d'arbre, l'intension des concepts précédée du nombre de documents appartenant à leur extension. Nous avons choisi d'afficher tous les concepts du treillis satisfiant les paramètres `Mnc` et `mnd`. Certains concepts apparaissent plusieurs fois en raison de l'héritage multiple dans le treillis et du choix d'affichage d'une hiérarchie dépliée. Pour chaque concept, l'utilisateur aura la possibilité de cliquer sur s'il estime que le concept est pertinent et sur s'il estime qu'il ne l'est pas.

– dans la partie droite de l'interface, une division est dédiée à l'affichage des documents associés à chacun des concepts. Le contenu de cette division est modifiée dynamiquement quand l'utilisateur passe la souris sur un des concepts de la hiérarchie. Tous les documents appartenant à l'extension du concept survolé sont affichés.

– dans la partie gauche, une division retrace l'historique des actions de l'utilisateur : requêtes soumises, évaluation de concepts pertinents et non pertinents. Dans le futur, cette division a pour objectif de permettre à l'utilisateur de revenir en arrière sur certaines de ses décisions, comme cela est présenté dans le moteur de recherche EXALEAD (`http://www.exalead.fr/`).

L'architecture de CRECHAINDO est donnée en Figure 5. CRECHAINDO interroge GOOGLE qui retourne une page HTML contenant une liste de documents tels

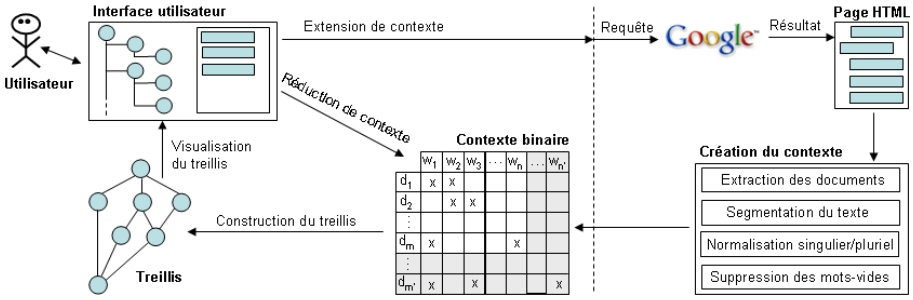

Figure 5. *L'architecture de* CRECHAINDO.

que celui présenté en Figure 3. Pour l'instant, la langue d'interrogation est obligatoirement l'anglais ; le résultat est constitué d'un ensemble de documents écrits en anglais. La page HTML est analysée par le système pour extraire l'ensemble des documents. Pour chaque document, le titre, l'extrait et l'URL sont extraits. Les nouveaux documents sont ajoutés à G_i (l'URL est utilisé comme clé pour éliminer les documents *doublons*). CRECHAINDO créé le contexte à partir de données textuelles selon l'approche classique d'indexation automatique proposée par (van Rijsbergen, 1979). Chaque document (titre et extrait) est segmenté en mots. Les variations flexionnelles de type singulier/pluriel sont traitées en utilisant les principales règles de variations de l'anglais : le `s` terminal (day → days, etc.), `an` → `en` (man → men, etc.), `y` → `ies` (baby → babies, etc.), `fe` → `ves` (wife → wives, etc.), etc. Étant donnés ces règles, le vocabulaire est normalisé en ne gardant que la forme la plus fréquente des mots apparaissant à la fois au pluriel et au singulier, pour éviter la dispersion du vocabulaire. Le treillis est d'ailleurs directement affecté par une consolidation de certaines extensions de concepts. Pour finir, nous utilisons une liste de mots vides récupérée du projet SNOWBALL (SNO 2007) pour éliminer les mots grammaticaux (comme *"a","the","of"*, etc.). Les mots restant sont utilisés pour créer le contexte *documents* × *mots* (il n'y a pas de sélection spécifique de mots). Une fois le contexte construit, CORON (Szathmary *et al.*, 2005) est utilisé pour calculer le nouveau treillis et une page HTML/javascript est générée.

4. Impact des interactions utilisateur sur la structure hiérarchique.

Cette partie présente une expérimentation et discute de l'intérêt des fonctionnalités offertes par CRECHAINDO. Supposons que l'utilisateur recherche des documents concernant les deux chercheurs italiens *Carpineto* et *Romano*, spécialistes de l'application des treillis à la RI. Ce type de recherche est très courant pour de la veille scientifique, pour l'analyse des travaux dans un domaine donné ou réalisés par des acteurs donnés. Soit *"carpineto romano"* la première requête soumise. La hiérarchie K_1, obtenue l'extension du contexte initial vide K_0 par $K_0 \xrightarrow{+\text{"carpineto romano"}} K_1$ est

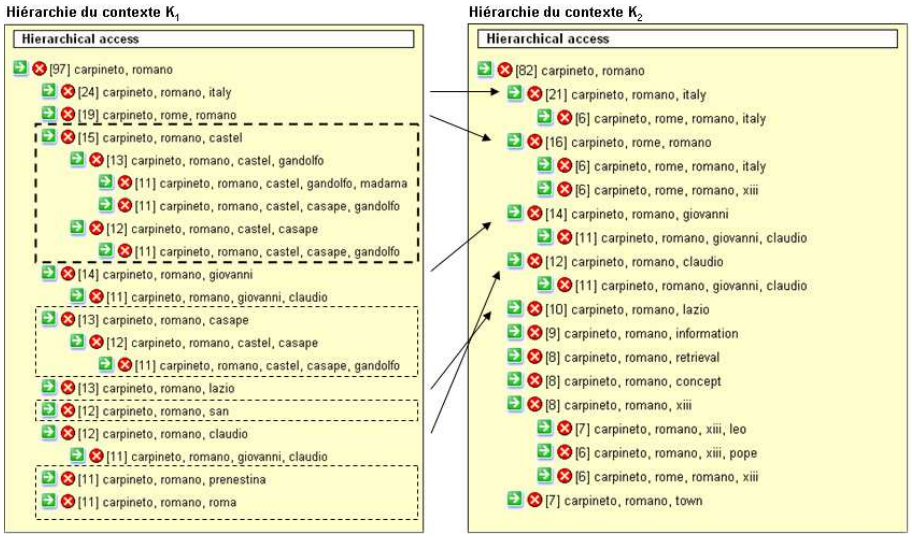

Figure 6. *De la hiérarchie de K_1 à la hiérarchie de K_2 par $K_1 \xrightarrow{-\text{"carpineto romano castel"}} K_2$.*

présentée en partie gauche de Figure 6. Nous pouvons noter que le concept le plus générique ne contient que 97 documents au lieu des 100 documents demandés. La raison est que 3 des 100 documents retournés par GOOGLE ne contiennent pas les 2 mots, ni dans le titre, ni dans l'extrait.

4.1. *Rejeter un concept non pertinent*

L'utilisateur peut rapidement constater que cette hiérarchie contient des concpets non pertinents pour une recherche concernant *Claudio Carpineto* et *Giovanni Romano*. Supposons que C, avec $intension(C) = \{carpineto, romano, castel\}$ soit un concept non pertinent. Cliquer sur ⊗ situé devant *"carpineto, romano, castel"* dans l'interface utilisateur produit la hiérarchie K_2 présentée en partie droite de la Figure 6. La hiérarchie K_2 ne contient maintenant plus que 82 documents. Dans une visualisation arborescente du treillis, l'effet obtenu est une suppression de branche enraciné en C, mais dans le treillis, les concepts de la branche sont soit détruits, soit *ajustés* si un héritage multiple les rattache à une autre partie du treillis. L'ajustement prend la forme d'une suppression, dans les concepts de la branche, des mots qui ont disparus du contexte suite à l'élimination des documents non pertinents (il s'agissait de mots uniquement présents dans les documents non pertinents). Éliminer des documents dans le contexte peut affecter l'ensemble de la hiérarchie. Comme illustré en partie gauche de la Figure 6, tous les sous-arbres encadrés par des pointillés ont été supprimés et

certaines extensions de concepts ont été réduites. Par exemple, 3 documents ont été supprimés dans l'extension du concept *"carpineto, romano, italy"*.

4.2. *Accepter un concept pertinent*

Soit C, tel que $intension(C) = \{carpineto, romano, giovanni, claudio\}$ un concept pertinent. Cliquer sur  situé devant C dans l'interface utilisateur produit la hiérarchie K_3 présentée en Figure 7. 87 nouveaux documents ont été ajoutés. Ils ont structuré plus finement la hiérarchie.

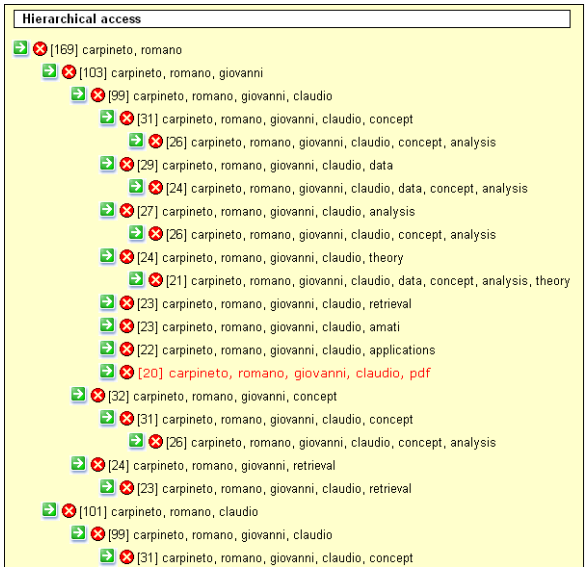

Figure 7. *Une partie de la hiérarchie de K_3, après $K_2 \xrightarrow{+\text{"carpineto romano giovanni claudio"}} K_3$.*

4.3. *Soumission d'une nouvelle requête*

Durant le processus de RI, l'utilisateur doit quelques fois reformuler son besoin par lui-même. Dans CRECHAINDO, chaque nouvelle requête soumise au système va contribuer à ajouter des documents au contexte. Par exemple, la partie de la hiérarchie présentée en Figure 8 est obtenue par extension de K_3 par une nouvelle requête sur *"carpineto romano lattice"* : $K_3 \xrightarrow{+\textit{"carpineto romano lattice"}} K_4$.

4.4. *Discussion*

Trois types d'interaction potentielle sont disponibles dans CRECHAINDO.

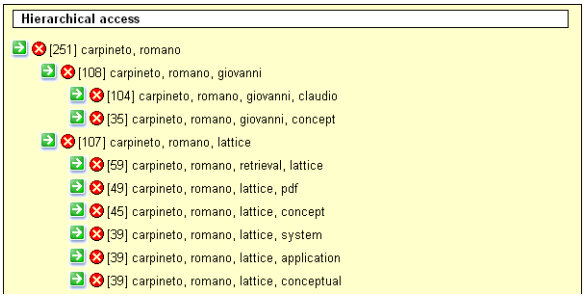

Figure 8. *Partie de la hiérarchie de* K_4*, après* $K_3 \xrightarrow{+\text{"carpineto romano lattice"}} K_4$.

Rejeter un concept non pertinent est très utile lorsque le treillis contient des concepts qui ne se focalisent pas immédiatement sur des informations satisfaisant l'utilisateur. Les concepts non pertinents sont souvent issus de requêtes avec des termes polysémiques, ou en raison d'une forte dispersion du vocabulaire dans les extraits de documents retournés par GOOGLE. Pour limiter ce problème, construire les concepts à partir d'un ensemble limité de propriétés — tel que cela est fait dans CREDO pour le premier niveau de la hiérarchie — est une solution possible. Mais dans ce cas, des documents et des concepts non pertinents sont toujours présents dans la hiérarchie. C'est pourquoi, il y a un réel besoin de *nettoyer* la hiérarchie. Et, éliminer le bruit permet indirectement de ce focaliser sur les documents pertinents.

Accepter un concept pertinent est un moyen d'introduire de nouveaux documents, plus spécifiques que ceux déjà obtenus jusqu'à présent, dans le contexte. Ceci est une extension significative de CREDO qui produit une hiérarchie limitée en profondeur et dont le degré de spécialisation des concepts est également limité. En effet, dans CREDO, le treillis est construit seulement sur 100 documents et le vocabulaire de ces documents ne couvre par nécessairement toute la variété de sujets en lien avec les mots employés pour interroger le système. De plus, l'extension de certains concepts peut être très petite, en raison de la distribution des 100 documents dans l'ensemble des concepts. Dans CRECHAINDO, si un concept C contient peu de documents — comme cela est le cas pour les concepts les plus bas dans la hiérarchie — accepter C étend la sous-hiérarchie de racine C. Des sous-concepts plus spécifiques que C seront générés par la construction d'un nouveau treillis. Ainsi, la profondeur de la hiérarchie et le degré de spécialisation sont moins limités.

Soumettre une nouvelle requête offre à l'utilisateur un moyen de reformuler son besoin par lui-même, sans être restreint par les concepts du treillis initial. Ainsi, l'utilisateur peut soumettre de multiples requêtes au système et toutes les réponses retournées par GOOGLE seront synthétisées dans une même hiérarchie. Cette approche semble intéressante pour fusionner les résultats provenant de multiples reformulations de requêtes et peut probablement être appliqué pour fusionner les résultats provenant

de multiples sources d'information comme cela est fait, par exemple, par un métamoteur de recherche.

4.5. *Perspectives*

Tout d'abord, l'efficacité de l'outil CRECHAINDO doit être évaluée et validée par rapport à l'accomplissement d'une tâche de RI. Plus généralement, nous pensons que de nombreux problèmes, concernant le processus de RI doivent être étudiés concrètement. Par exemple, quels sont les impacts de nos choix de modifications de contexte sur le processus de RI ? L'extension de contexte à partir de tous les mots formant l'intension d'un concept pertinent est peut-être trop réductrice : une extension de contexte à partir des mots spécifiques de l'intension du concept pertinent (i.e mots qui ne sont pas hérités des concepts génériques) peut être une alternative. De même, des alternatives existent pour la réduction de contexte qui actuellement se fait par élimination des documents rattachés à un concept non pertinent. Par exemple, une nouvelle interrogation de GOOGLE avec une requête composée de termes positifs (termes saisis par l'utilisateur) et de termes négatifs (termes spécifiques que l'utilisateur souhaite éliminer) pourraient enrichir l'ensemble des documents.

La *qualité* des hiérarchies, construites par différentes stratégies telles que celles proposées dans CREDO et CRECHAINDO, doit également être étudiée car celles-ci sont au cœur de ce type de processus de RI. Nous pensons également que certaines parties des treillis obtenus par CRECHAINDO suivent une organisation spécifique et que l'étude des propriétés du treillis est une façon d'améliorer la qualité de la hiérarchie. Par exemple, dans la Figure 7, le concept *"[99] carpineto, romano, giovanni, claudio"* est plus spécifique que *"[101] carpineto, romano, giovanni"* et que *"[103] carpineto, romano, claudio"*. Nous reconnaissons que l'héritage multiple favorise l'accès à un concept, qui est ainsi accessible par plusieurs chemins. Néanmoins, l'intérêt des deux concepts *"[101] carpineto, romano, giovanni"* et *"[103] carpineto, romano, claudio"* peut être discuté en raison de (1) leurs connections avec le concept *"[99] carpineto, romano, giovanni, claudio"* et de (2) la forte similarité de leurs extensions (contenant *[99], [101]* et *[103]* documents). La différence entre ces concepts découle principalement de l'exploitation des extraits de documents, dont les limites ne sont pas suffisamment larges pour couvrir les 4 mots *carpineto, romano, giovanni, claudio* (mais ces mots, absents des extraits, apparaissent dans les document intégraux).

Finalement, une direction de recherche attractive concerne le potentiel de synthèse des treillis sur un ensemble de résultats de recherche. Cette approche peut être appliquée pour une même requête soumise à plusieurs moteurs de recherche, ou également pour des requêtes multiples. Dans le cadre de CRECHAINDO, toutes les interactions utilisateurs qui sont stockées dans l'historique peuvent être exploitées pour du contrôle de pertinence implicite (Rieh *et al.*, 2006). À partir de l'ensemble des mots positifs (extraits des requêtes de l'utilisateur et des concepts pertinents) et de l'ensemble des mots négatifs (extraits des concepts non pertinents) plusieurs requêtes pourraient être

soumises à GOOGLE. Des requêtes peuvent être automatiquement générées par toutes les conjonctions possibles des mots positifs, avec un filtrage sur tous les mots négatifs.

5. Conclusion

Le système CRECHAINDO, presenté dans ce papier, est une application innovante de l'ACF pour la RI sur le web. Le contrôle de pertinence de l'utilisateur a été intégré à l'exploration d'un treillis structurant la liste de documents, initialement à plat, retournée par un moteur de recherche. Avec CRECHAINDO, l'utilisateur peut directement agir sur le treillis, pour le faire évoluer. Nettoyer le treillis, l'étendre dans une direction spécifique ou utiliser le treillis comme un outil de fusion pour des requêtes multiples offre des possibilités significatives pour la RI. Cette approche peut également être vue comme une approche de fouille du web, dans laquelle le résultat d'une étape est exploité dans l'étape suivante.

CRECHAINDO étend l'ACF pour la RI par un son aspect dynamique : le treillis évolue durant le processus de RI ; l'utilisateur n'est plus restreint à une structure statique calculée une fois pour toute.

6. Bibliographie

Carpineto C., Pietra A. D., Mizzaro S., Romano G., « Mobile Clustering Engine. », *Proceedings of the 28th European Conference on Information Retrieval, ECIR 2006*, vol. 3936 of *LNCS*, Springer, p. 155-166, 2006.

Carpineto C., Romano G., « Information retrieval through hybrid navigation of lattice representations », *International Journal of Computer Human Studies*, vol. 45, n° 5, p. 553-578, 1996.

Carpineto C., Romano G., « Effective Reformulation of Boolean Queries with Concept Lattices », *Flexible Query Answering Systems, Third International Conference (FQAS'98)*, vol. 1495 of *LNCS*, Springer, p. 83-94, 1998.

Carpineto C., Romano G., « Order-Theoretical Ranking », *Journal of the American Society for Information Science*, vol. 51, n° 7, p. 587-601, 2000.

Carpineto C., Romano G., « Exploiting the Potential of Concept Lattices for Information Retrieval with CREDO. », *Journal of Universal Computer Science*, vol. 10, n° 8, p. 985-1013, 2004.

Cole R., Eklund P., Stumme G., « Document Retrieval for Email Search and Discovery using Formal Concept Analysis », *Journal of Applied Artificial Intelligence (AAI)*, vol. 17, n° 3, p. 257-280, 2003.

Dmitriev P. A., Eiron N., Fontoura M., Shekita E., « Using annotations in enterprise search », *Proceedings of the 15th International Conference on World Wide Web*, p. 811-817, 2006.

Ducrou J., « DVDSleuth : A Case Study in Applied Formal Concept Analysis for Navigating Web Catalogs », *Conceptual Structures : Knowledge Architectures for Smart Applications, 15th International Conference on Conceptual Structures (ICCS 2007)*, vol. 4604 of *LNCS*, Springer, p. 496-500, 2007.

Ducrou J., Eklund P., « SearchSleuth : The Conceptual Neighbourhood of an Web Query », *in* , J. Diatta, , P. Eklund, , M. Liquiere (eds), *Fifth International Conference on Concept Lattices and Their Applications (CLA2007)*, p. 253-263, 2007.

Ducrou J., Vormbrock B., Eklund P. W., « FCA-Based Browsing and Searching of a Collection of Images », *Conceptual Structures : Inspiration and Application, 14th International Conference on Conceptual Structures (ICCS 2006)*, vol. 4068 of *LNCS*, Springer, p. 203-214, 2006.

Dumais S. T., Cutrell E., Sarin R., Horvitz E., « Implicit queries (IQ) for contextualized search (demo description) », *Proceedings of SIGIR 2004*, p. 594, 2004.

Ganter B., Wille R., *Formal Concept Analysis : Mathematical Foundations*, Springer, Berlin, 1999.

Kassab R., Lamirel J.-C., Nauer E., « Novelty Detection for Modeling User's Profile », *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference*, AAAI Press, California, p. 830-831, 2005.

Kelly D., Teevan J., « Implicit feedback for inferring user preference : a bibliography », *SIGIR Forum*, vol. 37, n° 2, p. 18-28, 2003.

Koester B., « Conceptual Knowledge Retrieval with FooCA : Improving Web Search Engine Results with Contexts and Concept Hierarchies. », *in* , P. Perner (ed.), *Advances in Data Mining, Applications in Medicine, Web Mining, Marketing, Image and Signal Mining, 6th Industrial Conference on Data Mining, ICDM 2006, Leipzig, Germany*, vol. 4065 of *Lecture Notes in Computer Science*, Springer, p. 176-190, 2006.

Rieh S. Y., Xie H., « Analysis of multiple query reformulations on the web : the interactive information retrieval context », *Information Processing and Management*, vol. 42, n° 3, p. 751-768, 2006.

Rocchio J., « Relevance feedback in information retrieval », *in* , G. Salton (ed.), *The SMART Retrieval System : Experiments in Automatic Document Processing*, p. 313-323, 1971.

Salton G., Buckley C., « Improving retrieval performance by relevance feedback », *JASIS*, vol. 41, n° 4, p. 288-297, 1990.

Shen X., Tan B., Zhai C., « Context-sensitive information retrieval using implicit feedback », *Proceedings of SIGIR'05*, 2005.

SNO, « Snowball project web site », http ://snowball.tartarus.org/, 2007.

Szathmary L., Napoli A., « CORON : A Framework for Levelwise Itemset Mining Algorithms », *Supplementary Proceedings of The Third International Conference on Formal Concept Analysis (ICFCA '05), Lens, France*, p. 110-113, 2005.

van Rijsbergen C. J., *Information Retrieval*, Butterworths, 1979.