Modélisation d'un système de recherche d'information pour les systèmes hypertextes. Application à la recherche d'information sur le World Wide Web

Fernando Jorge Carvalho de Aguiar

ÉCOLE NATIONALE SUPÉRIEURE
DES MINES DE SAINT-ETIENNE

UNIVERSITÉ JEAN MONNET
DE SAINT-ETIENNE

N° d'ordre : 272 ID

THÈSE

présentée par

Fernando Jorge CARVALHO DE AGUIAR

pour obtenir le grade de

Docteur

DE L'ÉCOLE NATIONALE SUPÉRIEURE DES MINES DE SAINT-ETIENNE ET
DE L'UNIVERSITÉ JEAN MONNET DE SAINT-ETIENNE

Spécialité Informatique

Modélisation d'un système de recherche d'information pour les systèmes hypertextes. Application à la recherche d'information sur le World Wide Web.

Soutenue publiquement le 28 juin 2002 devant le jury composé de :

Jacques SAVOY	Professeur à l'Université de Neuchâtel (Suisse)	Rapporteur / Pdt.
Patrick GALLINARI	Professeur à l'Université de Paris VI	Rapporteur
François JACQUENET	Professeur à l'Université Jean Monnet de Saint-Etienne	Examinateur
Jean-Pierre CHEVALLET	Maître de conférence à l'Université Pierre Mendès-France	Examinateur
Mohand BOUGHANEM	Maître de conférence à l'IRIT - Toulouse	Examinateur
Michel BEIGBEDER	Maître assistant à l'E.N.S. des Mines de Saint-Etienne	Examinateur
Jean-Jacques GIRARDOT	Maître de recherche à l'E.N.S. des Mines de Saint-Etienne	Examinateur

ÉCOLE NATIONALE SUPÉRIEURE
DES MINES DE SAINT-ETIENNE

UNIVERSITÉ JEAN MONNET
DE SAINT-ETIENNE

N° d'ordre : 272 ID

THÈSE

présentée par

Fernando Jorge CARVALHO DE AGUIAR

pour obtenir le grade de

Docteur

DE L'ÉCOLE NATIONALE SUPÉRIEURE DES MINES DE SAINT-ETIENNE ET
DE L'UNIVERSITÉ JEAN MONNET DE SAINT-ETIENNE

Spécialité Informatique

Modélisation d'un système de recherche d'information pour les systèmes hypertextes. Application à la recherche d'information sur le World Wide Web.

Soutenue publiquement le 28 juin 2002 devant le jury composé de :

Jacques SAVOY	Professeur à l'Université de Neuchâtel (Suisse)	Rapporteur / Pdt.
Patrick GALLINARI	Professeur à l'Université de Paris VI	Rapporteur
François JACQUENET	Professeur à l'Université Jean Monnet de Saint-Etienne	Examinateur
Jean-Pierre CHEVALLET	Maître de conférence à l'Université Pierre Mendès-France	Examinateur
Mohand BOUGHANEM	Maître de conférence à l'IRIT - Toulouse	Examinateur
Michel BEIGBEDER	Maître assistant à l'E.N.S. des Mines de Saint-Etienne	Examinateur
Jean-Jacques GIRARDOT	Maître de recherche à l'E.N.S. des Mines de Saint-Etienne	Examinateur

À Geraldo, Julieta,
Ayrton et Felipe.

Remerciements

Je tiens avant tout à exprimer mes plus vifs remerciements à tous les membres du jury :

Monsieur Jean-Jacques Girardot, maître de recherche et directeur du département RIM de l'École Nationale Supérieure des Mines de Saint-Etienne, pour m'avoir accueilli dans son département et pour la confiance qu'il m'a accordée en acceptant d'être directeur de cette thèse ;

Monsieur Jacques Savoy, professeur à l'Université de Neuchâtel (Suisse) pour l'honneur qu'il m'a fait d'accepter de présider le jury mais aussi pour avoir accepté d'être rapporteur de mon travail ;

Monsieur Patrick Gallinari, professeur et directeur adjoint du Laboratoire d'Informatique de Paris 6, LIP6, pour l'intérêt qu'il a accordé à mon travail en acceptant d'en être rapporteur ;

Monsieur François Jacquenet, professeur et directeur du département d'Informatique de l'Université Jean Monnet de Saint-Etienne, pour l'attention qu'il a bien voulu porter à ce travail en qualité d'examinateur ;

Monsieur Jean-Pierre Chevallet, maître de conférences à l'Université Pierre Mendès-France, Grenoble, pour avoir bien voulu examiner ce travail ;

Monsieur Mohand Boughanem, maître de conférences à l'Institut de Recherche en Informatique de Toulouse, IRIT, pour avoir accepté d'examiner ce travail ;

Monsieur Michel Beigbeder, maître assistant à l'École Nationale Supérieure des Mines de Saint-Etienne, pour avoir encadré et dirigé mon travail avec sérieux et sympathie. Je lui suis très reconnaissant du soutien qu'il a bien voulu m'apporter pendant toutes les étapes de ce travail ; cela a été fondamental dans la réussite de ce dernier.

Ma reconnaissance va aussi aux autres membres permanents de RIM pour leur sympathie : Jean-François, Philippe et Roland, ainsi qu'aux thésards et ex-thésards : Bich-Liên, Pierre, Faïza, Gildas, Bertrand et Yamina. Je tiens à remercier particulièrement Annie Corbel qui a toujours fait preuve de bienveillance envers moi.

J'aimerais aussi dire un grand merci aux permanents et aux thésards des autres départements des centres SITE et SIMMO avec qui j'ai eu le plaisir, entre autres, de discuter lors des pauses cafés et/ou déjeuners. Ces discussions d'une nature *extrêmement* intellectuelle

constituaient un repos fondamental dans mes journées de travail. Tout en prenant le risque d'oublier certains, je tiens à remercier : Natacha, Frédérique, Laurent, Saliha, Sophie, Bruno, Rose, Gilbert, Olivier, Frédéric, Franck, Thibault, Rémi et Jean-François.

Je ne saurais assez exprimer toute ma gratitude à Ana, Tarcísio, Eric, Tânia, Auriana, Alberto, Adriana, Guilherme, Karen, François, Luís et Cristina. De près ou de loin, leur soutien, présence, sagesse, encouragement et disponibilité ont constitué une ressource inépuisable de motivation grâce à laquelle j'ai pu mener à bien ce travail.

Enfin, mes parents et mes frères dont le soutien et la compréhension inconditionnels ont toujours été exemplaires.

Résumé

Dans un hypertexte, un document est souvent composé de plusieurs nœuds et non pas d'un seul. L'information véhiculée par un nœud donné peut difficilement être appréhendée à travers la lecture du seul contenu de ce nœud, le contenu des autres nœuds qui composent un document avec le premier nœud lui apportent un contexte. La connaissance de ce contexte est fondamentale dans la compréhension de l'information véhiculée par le premier nœud.

Un système de recherche d'information, ou plus couramment un moteur de recherche, appliqué au système hypertexte que constitue le Web devrait considérer dans son fonctionnement la fragmentation des documents hypertextuels en plusieurs pages : une page ne constitue pas un document à part entière, elle n'en est qu'une partie. Ainsi, pour bien indexer une page le contexte de l'information qu'elle véhicule doit être considéré. Les moteurs de recherche considérent souvent une page comme un document et l'indexent en analysant uniquement son contenu. Le contexte des pages est ignoré.

Dans ce travail nous proposons un modèle de recherche d'information pour un moteur de recherche appliqué à un système hypertexte constitué par un site Web. Ce modèle repose sur la construction d'un index à deux niveaux pour chacune des pages du site : un premier niveau, niveau inférieur, construit à partir du seul contenu de la page, et un deuxième niveau, niveau supérieur, construit à partir du contenu des pages qui apportent un contexte au contenu de la page en train d'être indexée. En améliorant la qualité des index des pages on cherche à améliorer l'efficacité du moteur de recherche.

Grâce à l'implémentation d'un prototype de moteur de recherche intégrant le modèle proposé ainsi que l'utilisation de la collection de tests WT10g issue des conférences TREC et adaptée à nos besoins, nous avons pu mener des expérimentations. Les résultats de ces dernières, une amélioration dans la qualité des réponses retournées par le moteur prototype, sont des indicateurs favorables de l'utilité de l'information contextuelle des pages. L'efficacité du moteur prototype a été comparée avec celle d'un moteur de recherche adoptant un modèle traditionnel où un seul niveau d'index, celui issu du seul contenu des pages, est utilisé.

Mots clés

Hypertexte, recherche d'information sur le Web, moteur de recherche, graphe du Web, analyse de liens, bibliométrie, classification automatique

A new model for hypertext information retrieval systems. Application to World Wide Web information retrieval.

Abstract

In a hypertext documents are seldom composed of a set of nodes instead of a single one. The information one page conveys might not be fully grasped if only the content of it is considered. The content of the pages with which the page being considered compose one document bear contextual information. Taking into account contextual information when indexing pages is fundamental to the quality of their index.

Information retrieval systems for the Web, commonly known as Web search engines, should consider the splitting up of Web documents into several pages: one page should not be considered as a fully-fledged document, it is only a part of it. Therefore, when indexing a page one should consider its contextual information which is seldom located in its neighborhood. Traditionally, Web search engines consider pages as fully-fledged documents and their index are then built only from their contents. Contextual information is not considered.

In this work we put forward a new information retrieval model for search engines running over Web sites. The cornerstone of it is a 2-level index for the pages composing the site: the bottom level is constructed solely from the content of the page itself, and the top level is constructed from the analysis of the contents of the pages which give a context to the page being indexed. We aim to improve the effectiveness of the search engine by improving the quality of the pages' index.

The implementation of a search engine prototype integrating the model suggested and the use of the test collection WT10g issued from the TREC conferences and adapted to our needs, allowed us to carry out a large number of tests. The results of these tests showed an improvement of the effectiveness of the search engine prototype when compared with that of a search engine integrating a traditional model where contextual information is not used to index pages. Therefore, the tests unveiled evidence that contextual information might be worth considering when modelling a search engine.

Keywords

Hypertext, Web information retrieval, search engine, Web graph, link analysis, bibliometrics, clustering

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Problématique

Dans la deuxième moitié du vingtième siècle, l'humanité a vécu l'apparition et la popularisation de l'ordinateur, des documents électroniques, des différents types de support pour stocker les documents et, finalement, des réseaux de télécommunications. Si nous considérons ces quatre événements comme un seul événement majeur — vu leur proximité dans le temps — nous pouvons affirmer sans trop risquer d'avoir tort qu'il s'agit de l'événement qui a affecté le plus la relation entre l'homme et l'information dans l'histoire de l'humanité après l'apparition de l'écriture et de l'imprimerie.

L'exploitation des capacités de ces nouvelles technologies est responsable de changements dans tous les niveaux du fonctionnement de la société moderne. Chacun peut constater des changements dans le contexte économique, les données politiques, le modèle social, les paramètres écologiques, les valeurs éthiques, les critères culturels et les attitudes individuelles.

De pair avec l'ordinateur, les documents électroniques ont été créés. Ils constituent la première façon de sauvegarder l'information sous une forme immatérielle. Les mémoires magnétiques, e.g., la bande magnétique, le disque rigide, le disque flexible, et les mémoires optiques, e.g., le vidéodisque, le cédérom, le cd-r, ont permis un stockage de données électroniques de très bonne qualité en termes de compacité et de pérennité pour un coût qui diminue sans cesse depuis leur création. Finalement, les réseaux informatiques ont rendu la diffusion et l'échange d'information plus simples que jamais.

L'irruption de l'internet au début des années 70 — la consolidation et la popularisation s'effectuant une dizaine d'années après — et ensuite l'apparition du Web[1] au début des années 90 comme l'un des *outils* mis en place grâce à l'Internet peuvent être considérés ensemble comme responsables d'un saut important dans une courbe imaginaire qui décrirait la quantité d'information facilement disponible et échangée entre les hommes au cours du temps.

La société moderne s'est ainsi offert une base propice à un développement accéléré. La pierre angulaire de cette base est sans doute la technologie de création, de stockage et de

[1] *World-Wide Web*, en français : la toile d'araignée mondiale.

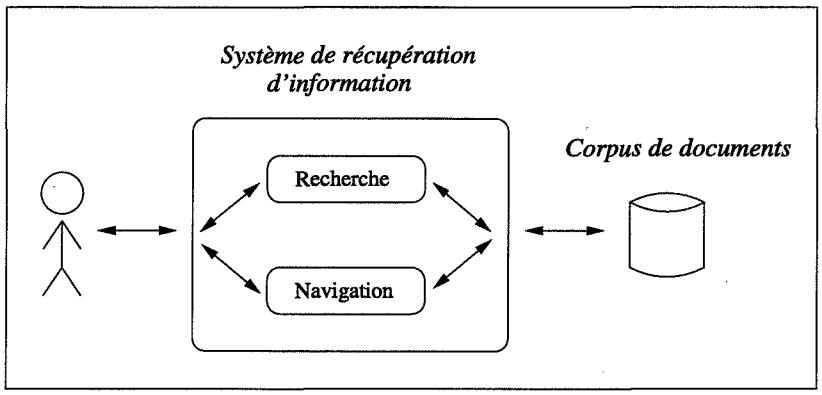

FIG. 1.1: Modèle traditionnel d'un système de récupération d'information.

diffusion de l'information. Mais toutes ces conditions réunies ne sont pas encore suffisantes pour que ce développement soit complètement valorisé. En effet, il faut qu'existent des outils pour aider l'homme dans le processus de consultation de la masse d'information accessible, car la situation où l'on ne réussit pas à récupérer une information que l'on sait existante et accessible ne diffère pas énormément de celle où l'information n'existe pas.

Afin de faciliter le processus de récupération d'information, nombre de communautés scientifiques coopèrent dans l'étude et la mise en place de *systèmes de récupération d'information* dont la tâche est d'aider l'utilisateur dans les processus de localisation et consultation d'information. On peut ranger ces systèmes en deux grandes classes selon le type de mécanisme qu'ils mettent en place dans le processus de récupération. Les types de mécanisme sont la recherche et la navigation[2]. Dans le mécanisme de recherche l'utilisateur exprime l'information qu'il souhaite récupérer sous la forme d'une requête, p.ex. une liste de termes, et le système lui retourne la liste des documents qu'il a jugés comme étant les plus pertinents à la requête. Dans le mécanisme de navigation l'utilisateur parcourt la masse ou le corpus d'information d'une manière plus ou moins aléatoire. C'est un mécanisme qui peut s'avérer efficace dans les cas où le besoin d'information de l'utilisateur n'est pas bien défini et où il est naturellement trop vague.

Le choix du type de mécanisme de récupération de l'information à utiliser dans un système doit tenir compte des caractéristiques de l'information contenue dans le corpus. Par exemple, si le corpus contient des documents complètement autonomes, i.e. n'ayant pas de relations explicites avec d'autres documents du corpus, un système de récupération d'information basée sur la navigation serait naturellement inadéquat puisque l'exploitation des documents se ferait d'une manière complètement aléatoire. Par contre, si le corpus est constitué de documents organisés d'une manière hypertextuelle, un système de récupération basé sur la navigation peut s'avérer plus efficient que le mécanisme de recherche. Le niveau d'efficience dépendra entre autres de la qualité de la structure reliant les documents.

Notre travail s'insère dans le contexte de la récupération d'information dans les systèmes

[2]En anglais *browsing*.

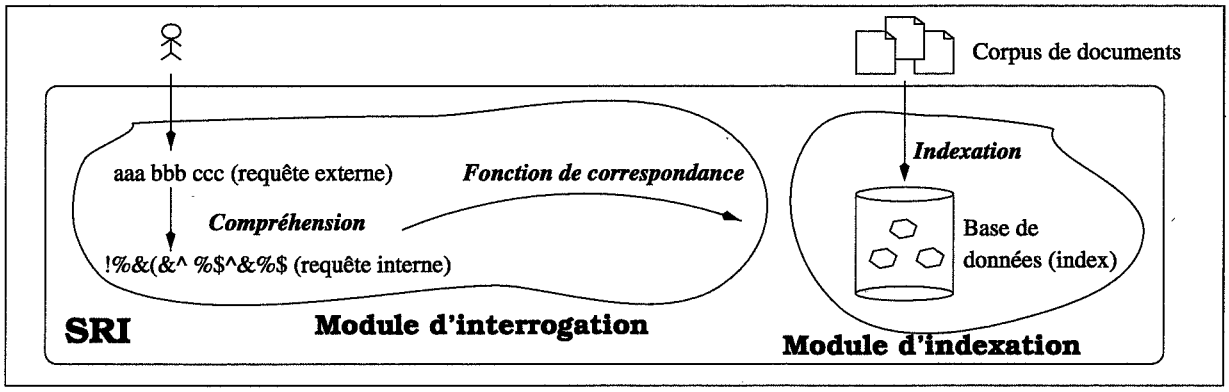

FIG. 1.2: Modèle d'un système de recherche d'information.

hypertextes. En particulier, nous nous intéressons à la récupération d'information sur le Web. Dans ce cadre, la situation constatée actuellement est très proche de celle décrite plus haut, à savoir que l'information recherchée par un utilisateur peut exister et être disponible dans le Web mais que sa récupération constitue souvent une tâche très difficile. Parmi les innombrables facteurs qui contribuent à cette difficulté nous pouvons citer : le volume d'information qui croît d'une façon exponentielle, son hétérogénéité, le manque d'organisation dans le stockage et la diffusion de l'information, etc.

Nous supposons que (i) le mécanisme de navigation tout seul n'est pas adéquat pour la récupération d'information sur le Web et que (ii) le mécanisme de recherche actuellement appliqué par les systèmes de recherche d'information — communément appelés *moteurs de recherche* dans le Web — n'est pas adéquat non plus puisqu'il ne prend pas en compte les particularités de l'information hypertextuelle disponible dans le Web.

Dans ce qui suit nous définissons brièvement les systèmes de recherche d'information. Ensuite nous abordons les systèmes hypertextes et leur mécanisme de récupération d'information. Finalement, nous précisons les contributions de notre travail et l'organisation de ce rapport.

1.2 Systèmes de recherche d'information

Un système de recherche d'information, SRI, est un système qui s'appuie sur le mécanisme de recherche (figure 1.1) pour effectuer la tâche de récupération d'information. Un SRI a pour but de satisfaire le besoin d'information d'un utilisateur exprimé à travers une requête. Le système doit ainsi retourner à l'utilisateur le maximum de citations à des document pertinents à la requête et le minimum de citations à des documents non-pertinents. La figure (figure 1.2) illustre le modèle basique d'un système de recherche d'information.

Nous pouvons considérer un système de recherche d'information comme étant composé par deux modules principaux : le module d'indexation et le module d'interrogation. Dans ce qui suit, nous analysons d'abord le module d'indexation et, ensuite, le module d'interrogation.

1.2.1 Module d'indexation

Bien entendu tout système de recherche d'information effectue une recherche restreinte à un corpus documentaire donné. Avant d'aborder le processus d'indexation il est important de signaler la difficulté pour trouver une définition précise pour le terme document dans le contexte de la recherche d'information. Les définitions trouvées dans les dictionnaires sont presque toujours très générales — ce qui est normal — et font référence à une *chose écrite qui sert à renseigner, à prouver.* Par exemple, dans le Petit Robert nous trouvons la définition suivante : « *écrit, servant de preuve ou de renseignement.* » Dans cette étude nous avons opté par une définition liée au contenu d'un document :

Définition 1 *Un document est un volume d'information auto-explicative.*

Par *information auto-explicative* nous entendons une information qui ne dépend pas d'une autre pour être comprise. Par exemple, un livre constitue un document parce que le volume d'information qu'il renferme est très souvent suffisant pour qu'un lecteur comprenne ce que l'auteur a voulu exprimer avec son ouvrage. Par contre, si nous prenons un paragraphe de ce livre au hasard, nous ne pouvons en général pas appréhender l'information véhiculée par le paragraphe sans avoir lu le chapitre qui contient le paragraphe, voire le livre entier. En effet, il manque le contexte de l'information. Nous parlerons davantage de la notion de contexte dans le chapitre 3.

Le processus principal du module d'indexation est le processus d'indexation du corpus. Ce processus consiste à créer un index ou un représentant pour chaque document composant le corpus documentaire en question. Définissons ces notions :

Définition 2 *Un index est une représentation synthétique du contenu sémantique d'un document.*

Définition 3 *L'indexation est le processus responsable de l'extraction du contenu sémantique d'un document et de la représentation de ce contenu sous la forme d'un index.*

Un index est représenté sous un format donné. Le formalisme qui permet de représenter un index est appelé le langage d'indexation. La complexité de ce formalisme peut être celle associée à un simple ensemble de termes et aller jusqu'à celle associée aux graphes conceptuels où les nœuds représentent des concepts et les arcs entre les nœuds représentent les relations existant entre les différents concepts.

Le contenu sémantique d'un document peut être extrait selon différentes techniques. À l'instar des langages d'indexation, les techniques d'indexation peuvent aussi être plus ou moins complexes[3] selon le SRI analysé. Pour ce qui est du texte, nous pouvons imaginer le processus d'extraction de contenu sémantique comme consistant d'abord en un décompte du nombre d'apparitions des termes dans un document donné, puis les termes dont les fréquences sont les plus élevées sont choisis comme étant les représentants du document en question. Ces idées sont originaires des travaux de Luhn [Luh57] publiés au début des années 60. À l'autre opposé

[3] Complexe dans le sens de difficile à comprendre, compliqué, et non pas dans le sens algorithmique du terme.

en termes de complexité, nous pouvons citer les algorithmes qui appliquent des techniques de traitement du langage naturel pour extraire le contenu sémantique des documents. Il est important de souligner le fait que la totalité des systèmes de recherche d'information grand-public disponibles sur le Web — communément appelés moteurs de recherche — utilisent des algorithmes d'indexation basés sur une analyse statistique de la fréquence d'occurrence des mots dans les pages HTML, considérés par la plupart des moteurs de recherche comme des documents.

Finalement, l'algorithme et le langage d'indexation utilisés par un SRI constituent ce que l'on appelle couramment le modèle d'indexation du SRI. En fait, on fait souvent référence à un modèle d'indexation et non pas à ses deux composants séparément puisque ceux-ci sont très dépendants entre eux, i.e. au niveau de la modélisation ce sont deux composants qui se complémentent. En effet, on ne peut que difficilement remplacer le langage d'indexation en gardant l'algorithme d'indexation.

1.2.2 Module d'interrogation

Le deuxième module d'un SRI est le module d'interrogation. Il traite les interactions entre l'utilisateur et le système. Ce module a deux tâches principales : la compréhension de la requête posée par l'utilisateur et l'évaluation de la fonction de correspondance entre la requête et les index des documents contenus dans la base de données du système.

La compréhension de la requête est une étape nécessaire. En effet, pour appliquer la fonction de correspondance, il faut que la requête et la représentation des documents, c'est-à-dire les index, soient représentées sous un même formalisme. Or, selon le SRI, la requête de l'utilisateur peut ne pas avoir le même format que les index. Dans ce cas, on peut imaginer le processus de compréhension comme une traduction d'une requête externe en une requête interne.

La deuxième tâche du module d'interrogation consiste à évaluer la pertinence de chaque index contenu dans la base à la requête interne. La fonction de correspondance[4] est chargée de quantifier cette pertinence. Finalement, le système rend à l'utilisateur une liste de citations de documents, dans la plupart des cas triées en ordre décroissant de pertinence, que la fonction de correspondance a jugés pertinents à la requête interne.

Analysons soigneusement cette étape afin d'insister sur trois points cruciaux dans le fonctionnement d'un SRI. Premièrement, ce sont les index des documents qui sont jugés par rapport à une requête et non pas les documents eux-mêmes. Ainsi, pour être efficient, il faut que le système assure que les index représentent bien le contenu sémantique des documents du corpus. C'est la première grande difficulté d'un système de recherche d'information comme on l'a souligné plus haut. Deuxièmement, les index sont comparés avec la requête interne et non pas avec la requête formulée par l'utilisateur. Or, si le système n'assure pas une traduction correcte de la requête externe, le système répondra à une requête exprimant un besoin d'information qui ne correspond peut-être pas à celui exprimé par l'utilisateur lors de la formulation de sa requête. Encore faut-il que l'utilisateur soit capable d'exprimer son besoin

[4] Ou fonction d'appariement.

d'information dans le langage de requête proposé par le système ce qui n'est pas toujours très facile. Mais cela est plutôt une difficulté du côté de l'utilisateur et non pas du système lui-même. Finalement, même si le système assure l'efficience des processus d'indexation et de traduction de la requête, il faut savoir utiliser correctement les index pour être capable de trancher sur la pertinence ou non des documents à une requête donnée. Il est question donc de la qualité de la fonction de correspondance.

Bien entendu, la modélisation du module d'indexation n'est pas indépendante de celle du module d'interrogation. En effet, lors de la conception d'un SRI, un modèle général est adopté. Ce modèle général définit le langage de requête, le modèle d'indexation et la fonction de correspondance. Les principaux types de modèles sont le modèle booléen, le modèle vectoriel [Sal71] et le modèle probabiliste [Rob77]. Nous n'allons pas rentrer dans les détails de chacun de ce modèles ici puisque leur analyse ne concerne pas directement ce travail. Nous nous contentons de donner les références aux travaux pionniers concernant chacun de ces modèles. En outre, une très bonne explication des modèles principaux ainsi que des modèles alternatifs est disponible dans [BYRN99]. Dans notre travail, nous nous servons d'un de ces modèles, en l'occurrence le modèle vectoriel, pour valider nos idées. Dans le chapitre 3 nous rappelons brièvement les concepts fondamentaux de ce modèle.

1.3 Systèmes hypertextes

Un système hypertexte ou, tout simplement, un hypertexte est un volume d'information disponible sous forme électronique dont la structure n'est pas linéaire. Le volume d'information peut être considéré comme un ensemble de morceaux d'information (les nœuds). Dans une structure hypertextuelle, chaque morceau peut être relié à un ou plusieurs morceaux par des *liens hypertextuels*. Les liens permettent d'exploiter le volume d'information, i.e. parcourir les différents nœuds de plusieurs façons, contrairement à une structure linéaire qui privilégie fortement un seul parcours.

Il est pourtant vrai que l'on peut aussi imaginer un livre comme étant un système hypertexte. En effet, l'information contenue dans un livre n'a pas toujours une structure linéaire. Par exemple, l'index ou la table de matières d'un livre permettent une exploitation non-linéaire du contenu d'un livre. Ce n'est pas pour autant qu'un livre constitue un hypertexte. Il y a deux raisons à ceci. Premièrement, les liens entre les termes de l'index et les passages dans le corps du texte sont *basés sur la syntaxe, i.e. sur des chaînes alphanumériques*. Or, les liens dans un système hypertexte sont plus riches puisqu'ils relèvent de la sémantique de l'information. Quelques exemples de types de liens hypertextes sont *problème-solution*, *tout-partie*, *cause-effet* et *lien chronologique* [Tea95].

Ainsi la diversité et la complexité des associations entre deux morceaux d'information que l'on peut trouver dans un hypertexte n'est pas la même que celle des associations qu'on trouve dans un livre ordinaire. Toutefois, nous pouvons toujours imaginer un super livre ayant des outils annexes comme des index qui permettraient la mise en place de liens exprimant des associations très riches.

Finalement, ce qui démarque la frontière entre un livre, fût-il associé à des associations complexes et diverses, et un hypertexte est le média sur lequel est disponible l'information, en l'occurrence le média électronique. La caractérisation de l'hypertexte donnée par Kinnel en 1992 [KF92] est très claire à ce propos : « *...des liens électroniques sont créés entre morceaux d'information d'un document codé dans un format compréhensible par un ordinateur. L'ordinateur se charge des sauts entre ces morceaux d'information liés. L'ordinateur est fondamental non seulement pour la définition des liens pendant la création d'un document hypertexte mais aussi pour l'activation de ces liens lors de l'exploitation du document...* »

En effet, l'informatique ouvre d'autres horizons aux types de fonctionnalités envisageables pour une organisation non-linéaire de l'information. Un système hypertexte peut être utilisé comme un outil de travail interactif et partagé, un didacticiel, etc. Pourtant son utilisation principale reste l'exploitation de l'information. Étant donné qu'un système hypertexte permet une structuration très riche d'un volume d'information, les manières d'exploiter l'information deviennent aussi très variées. Dans ce qui suit, nous rappelons brièvement l'histoire du concept d'hypertexte et, ensuite, nous discutons l'exploitation de l'information dans un tel système.

1.3.1 Histoire de l'hypertexte

Les principes du concept de l'hypertexte prennent leur origine dans l'article intitulé *As we may think*[5] [Bus45] de Vanevar Bush publié en 1945. L'hypertexte n'est donc pas un concept récent. Dans cet article, le système imaginaire *Memex* est présenté. Parmi les fonctionnalités de *Memex* nous retrouvons la création de documents ayant une structure non-linéaire, la lecture de ces documents suivant différents parcours, la mise en relation des documents suivant la nature associative de l'esprit humain et l'annotation des documents. Comme la technologie disponible à l'époque ne permettait pas le développement d'un tel système, Bush s'est contenté d'en décrire les fonctionnalités. Ce n'est que 20 ans plus tard que le terme *hypertexte* a été créé par le chercheur Theodor Nelson. En 1985, avec la commercialisation du logiciel d'édition d'hypertextes *Hypercard*, les systèmes hypertextes commencent à se populariser. Son usage s'est un peu banalisé avec l'apparition de bornes d'orientation et de consultation dans les lieux publics. Dans ces applications, l'information est souvent organisée d'une manière hypertextuelle. Aussi, plusieurs applications disponibles sur cédérom s'appuyaient sur cette technologie. La première conférence internationale sur les hypertextes a eu lieu en 1987 [ACM87].

Cependant, ce n'est qu'avec l'apparition du Web en 1992 que l'étude des hypertextes a gagné une certaine notoriété. L'utilisation du réseau *Internet* a permis la construction d'un hypertexte — du moins des morceaux d'information reliés — complètement décentralisé et universel. Son grand succès est peut-être dû à la liberté totale de publication qui lui est caractéristique. N'importe qui peut publier n'importe quoi (pas de censure) et n'importe comment (pas de ligne éditoriale). En revanche, cette même liberté de publication constitue la cause principale de la difficulté à laquelle on est confronté actuellement pour retrouver de l'information dans le Web. Par ailleurs, certains chercheurs affirment même que le Web ne constitue pas, stricto sensu, un système hypertexte du moment qu'il n'y a pas de charte définissant l'organisation des nœuds, l'affichage des données et le modèle de données[6]. Cette étude concerne

[5] En français, *Comme nous pouvons penser.*

[6] Nous sommes plutôt de cet avis là, toutefois nous n'allons pas approfondir cette question, d'autant plus que

la recherche d'information sur le Web. Les données qui y sont disponibles — que le Web soit considéré comme un système hypertexte ou non — ont certaines particularités qui doivent être prises en compte lors de la modélisation d'un système de recherche d'information. Dans cette étude nous considérons le Web comme étant un ensemble de ressources faiblement couplées ; certaines de ces ressources pouvant être considérées comme des systèmes hypertextes locaux dans la mesure où leur conception relève d'une certaine cohérence. Par ressource nous entendons un ensemble de pages regroupées dans ce que l'on appelle communément *site Web*. Dans la section 3.1 nous détaillerons ce que nous entendons par site Web.

1.3.2 La récupération d'information dans un système hypertexte

Nous avons mentionné plus haut qu'un système hypertexte pouvait avoir plusieurs fonctionnalités, mais que la principale était liée à l'exploitation de l'information. La façon d'exploiter l'information qui est inhérente à un système hypertexte est la navigation. Nous définissons déjà en quoi consiste la navigation et ensuite nous abordons le type de navigation effectué dans un système hypertexte.

La navigation

La navigation dans un ensemble de documents consiste à les parcourir et les analyser dans un ordre qui n'est pas défini a priori, voire qui est aléatoire. Par exemple, imaginez qu'une personne veut trouver un livre dans une bibliothèque où les livres sont déposés aléatoirement dans les étagères. Si la bibliothèque ne dispose pas de système qui peut indiquer à la personne l'emplacement du livre dans la bibliothèque, sa seule option est de parcourir les documents un par un afin de trouver celui qui est souhaité. La navigation est aussi utile dans les cas où l'utilisateur a un besoin d'information qui n'est pas très bien défini ou lorsque l'utilisateur veut plutôt explorer l'ensemble de documents disponibles et non pas trouver un document précis.

Les types de navigation

L'exemple de la bibliothèque montre une navigation sur un volume d'information organisé d'une manière plate. En fait, la seule organisation que l'on puisse imaginer dans cette situation est celle d'une liste[7] de livres. Or, pour effectuer une navigation un peu plus efficace, il faudrait que le volume d'information possède une structure plus complexe que celle représentée par une liste.

Dans le but de permettre une meilleure organisation de l'information, des classifications ont été créées. On ne peut pas ne pas citer la classification décimale de Dewey, CDD, publiée pour la première fois en 1876 et largement utilisée jusqu'à aujourd'hui dans de nombreuses bibliothèques à travers le monde. Il s'agit d'une répartition de l'ensemble de connaissances en catégories de domaines de connaissance organisées hiérarchiquement. En se basant sur la CDD, le belge Paul Otlet créa en 1905 la classification décimale universelle, CDU. Son originalité par rapport à la CDD réside dans la possibilité de combiner deux catégories différentes pour classifier une œuvre. D'ailleurs, plusieurs chercheurs affirment que ce n'est pas Bush le premier a avoir pensé à ce que Nelson appellerait en 1965 l'hypertexte, mais que c'est bel

le concept d'hypertexte n'a pas de définition acceptée par toutes les communautés qui travaillent dans des domaines liés.

[7] D'où le terme organisation plate.

et bien Otlet. En effet, à l'aide de fiches de carton d'une taille fixe contenant chacune des informations sur un ouvrage, Otlet a peut-être mis en place le premier hypertexte. Les fiches peuvent être considérées comme étant les nœuds de l'hypertexte, les liens étant assurés par la structure de la CDU.

La navigation s'appuyant sur la structure d'une classification comme celle de la CDD, i.e. une hiérarchie, permet une récupération ou exploitation plus efficace de l'information. Cependant, une hiérarchie constitue une structure très rigide pour la classification de documents. Un document ne peut être associé qu'à une seule classe, i.e. à un seul nœud de la hiérarchie. Imaginez un utilisateur en train de se servir de la CDD pour retrouver un document dans une bibliothèque. Le fait que l'œuvre est associée à un seul nœud de la hiérarchie sous-jacente à la classification implique qu'il existe un unique chemin à emprunter dans la hiérarchie pour retrouver la catégorie dans laquelle le livre se trouve. Supposez maintenant qu'un livre soit lié à deux thématiques différentes situées dans deux sous-arbres différents de la hiérarchie. Le livre doit être placé dans un unique nœud de la hiérarchie. Avec la CDD, le livre sera placé sous une thématique aux dépens de l'autre thématique. Si le raisonnement du lecteur qui cherche le livre en traversant la hiérarchie se focalise sur la thématique qui n'a pas été utilisé dans le classement, la recherche échouera.

Avec la fonctionnalité de combiner deux ou plusieurs catégories de la CDU, Otlet a attaqué le problème cité dans le dernier paragraphe. La combinaison de deux catégories peut être considérée comme la création d'une nouvelle classe dans la classification. Ainsi, le raisonnement ou le cheminement d'associations qui mène le lecteur du haut de la classification vers la classe à laquelle le document en question a été associé peuvent être multiples. Nous pouvons considérer que dans une telle structure, une information peut être vue ou considérée de manières multiples. En termes de structure, la CDU s'approche davantage de la structure d'un système hypertexte que la CDD. En effet, alors que la structure de la CDD représente une hiérarchie ou un arbre, la structure retrouvée dans la CDU est un graphe.

À l'instar des classifications, un système hypertexte offre aussi une navigation organisée dans la mesure où l'information est organisée selon une certaine structure. La structure utilisée est celle d'un graphe où les nœuds correspondent aux morceaux d'information et les liens correspondent à des relations entre les morceaux d'information. D'après cette définition, nous pouvons certes considérer le système développé par Otlet et basé sur la CDU comme étant un hypertexte[8]. Mais il s'agit d'un hypertexte simplifié dans la mesure où les relations entre les différents nœuds sont d'un même type, ont la même sémantique. En fait, dans les classifications, les liens ont, dans la plupart des cas, une sémantique de généricité/spécificité. Dans un système hypertexte il n'y a pas de limitation au niveau des types de liens. *L'hypertexte représente vraisemblablement la liberté maximale en termes de capacité de structuration.*

Remarquons enfin que l'efficacité de la navigation comme technique de récupération d'information au sein d'un hypertexte dépend beaucoup de sa structure. Nous examinons cette question dans la suite.

[8] Si l'on oublie la composante informatique, fondamentale dans l'hypertexte.

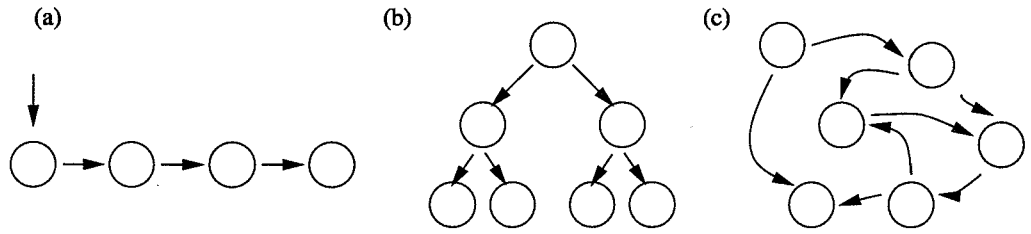

FIG. 1.3: Trois types de structure : (a) une liste, (b) une hiérarchie, (c) une structure hypertextuelle.

1.3.3 L'efficacité de la navigation

Dans son article, Bush critique durement les systèmes d'indexation de l'époque : « *Notre inaptitude à retrouver un document est largement due au caractère artificiel des systèmes d'indexation...* » C'est dans ce contexte que s'insère la notion des pistes associatives du système hypertexte suggéré dans son fameux article. Une piste associative met en relation plusieurs morceaux d'information dans une structure qui simule le cheminement de la pensée. Bush prônait que si l'homme raisonne et crée de nouvelles informations à travers un processus d'association, une structuration de l'information reflétant les associations d'idées permettrait une consultation et récupération plus efficace et efficiente de l'information. Les pistes associatives servent donc comme des guides pour un utilisateur en quête d'information dans un énorme univers de données. D'un point de vue de structuration de l'information, l'idée des pistes associatives est très intéressante dans la mesure où un morceau d'information peut faire partie de plusieurs pistes associatives en même temps et ainsi être considéré ou consulté de différentes manières.

Toutefois, si l'on se situe du côté de la récupération de l'information, on peut signaler quelques inconvénients à l'utilisation des pistes associatives. Premièrement, une piste associative relève des associations faites par un individu qui, à un instant donné, a considéré que les morceaux d'information constituant la piste s'associaient d'une certaine manière. Or, l'association entre deux informations est basée sur la formation et les intérêts de cet individu. De ce fait, toute association relève d'un jugement personnel. Par conséquent, les jugements d'association qui sont pertinents à un individu peuvent sembler non-pertinents à un autre.

Supposons maintenant qu'un utilisateur soit en train de chercher une information et que le processus d'association qui lui est pertinent corresponde à une piste associative basée sur les associations qui lui semblent pertinentes. Allons encore plus loin, (i) supposons qu'à tous les besoins d'information imaginables correspondent des pistes associatives déjà créées dans l'hypertexte. Il reste à l'utilisateur à trouver la bonne piste associative. Si l'on n'indique pas à l'utilisateur un des composants de cette liste associative il lui faudra le trouver. À ce moment, nous pouvons nous demander si un index de pistes associatives ne serait pas envisageable. En effet, certaines fois on peut identifier un index de pistes associatives dans le nœud par lequel débute en quelque sorte l'exploration de l'hypertexte, i.e. le nœud qui est affiché lorsque l'hypertexte est lancé. Nous pouvons alors (ii) supposer qu'un système hypertexte dispose toujours d'un index de pistes associatives dont l'utilisateur peut se servir pour retrouver la

piste qui le mènera vers l'information recherchée.

Malgré ces deux suppositions assez invraisemblables, l'utilisateur rencontre encore de grandes difficultés pour récupérer une information dans un hypertexte. Ces difficultés sont dues au processus de navigation même. Très souvent, avant d'activer un lien on ne peut pas savoir ce que l'on va trouver derrière lui. Autrement dit, il manque une grammaire qui permette de spécifier les types d'enchaînement qu'un lien peut apporter lorsqu'il est activé. La navigation effectuée un peu à l'aveuglette mène l'utilisateur à deux types principaux de désorientation cognitive : « *le problème du musée d'art* » [Tea95] et « *les digressions imbriquées* » [LC95]. Le premier type de désorientation est dû à la richesse même de la structure non-linéaire de l'information. L'utilisateur voit tellement d'informations et d'associations entre les informations qu'il finit par ne pas appréhender *l'essentiel* de l'information. C'est *l'indigestion intellectuelle.* L'homme est beaucoup plus habitué à la lecture d'une information organisée d'une façon linéaire, e.g. un roman, ou hiérarchique, e.g. un livre technique comportant de sections, sous-sections, etc., qu'à la lecture d'une information disposée de manière perçue comme anarchique comme c'est le cas de celle accessible à travers un système hypertexte.

Le deuxième type de désorientation est celui nommé « *les digressions imbriquées* » par Le Crosnier. À force de suivre des cheminements marginaux dans la partie de l'hypertexte qu'il est en train d'exploiter, l'utilisateur finit par ne plus savoir ni ce qu'il était en train de chercher au début de l'exploitation ni ce qu'il est en train de chercher à présent.

Finalement, si nous considérons un système hypertexte comportant un nombre conséquent de nœuds nous pouvons prévoir que même une navigation effectuée dans des conditions parfaites (index de pistes associatives, types des liens explicites et minimisation des désorientations cognitives) sur cet hypertexte ne conviendrait pas à un utilisateur qui n'a pas le temps ni l'envie de s'investir dans une traversée de l'hypertexte afin de récupérer l'information souhaitée. Encore pire, dans la plupart des cas, l'utilisateur n'est même pas sûr que l'information qu'il recherche est disponible dans l'hypertexte qu'il exploite. Ainsi, il peut se lancer dans une traversée de l'hypertexte et, à la fin du processus, conclure que l'information souhaitée n'y est pas disponible.

Pour résumer, les problèmes liés (i) aux pistes associatives, (ii) aux liens sans sémantique précise, (iii) à un volume important de l'hypertexte et (iv) aux problèmes intrinsèques de la technique de navigation font que cette technique de récupération d'information n'est peut-être pas toujours la plus adéquate pour exécuter la tâche de récupération d'information dans un système hypertexte. Une méthode d'accès plus direct à l'information peut être très utile dans certaines situations. La mise en place d'un système de recherche d'information sur un système hypertexte semble être très utile et complémentaire à la technique de navigation pour la tâche de récupération d'information.

Ainsi, nous tenons à souligner le fait que nous ne prônons pas un remplacement de la technique de navigation par la technique de recherche. Nous croyons que la navigation est un paradigme très intéressant pour la tâche de récupération d'information. Toutefois, il ne peut pas à lui seul assurer une exécution efficace de cette tâche. Par contre, le paradigme de *recherche* peut en effet rapidement amener l'utilisateur dans le voisinage des nœuds qui

l'intéressent. Ensuite, l'utilisateur peut se servir de la navigation pour exploiter ce voisinage et finalement trouver l'information recherchée.

1.4 Contribution de la thèse

L'objectif de ce travail est de proposer un modèle de système de recherche d'information adapté à un système hypertexte accessible par le Web. Nous nous intéressons aux systèmes hypertextes représentant les sites Web. Il n'existe pas de définition précise de ce qui constitue un site Web. Dans cette étude nous considérons un site Web comme étant un ensemble de pages associées à une thématique commune, maintenues par une seule personne ou un groupe de personnes, et **publié comme un ensemble cohérent d'information.** De plus, l'ensemble de pages d'un site est normalement hébergé par un serveur HTTP unique.

La particularité du modèle est de tenir compte des différents niveaux de granularité d'information pouvant être associés aux pages HTML. Notre hypothèse de base est qu'une page ne correspond pas nécessairement à un document tel que nous l'avons défini plus haut. L'information qu'une page véhicule ne peut pas être saisie à partir uniquement du contenu de la page. Il existe de l'information qui contextualise et donne un sens au contenu d'une page. L'information contextuelle d'une page donnée se trouve souvent dans des pages situées dans son voisinage. Nous appelons ces dernières pages des pages complémentaires. Une première contribution de cette thèse consiste dans la proposition d'une méthode pour trouver les pages complémentaires d'une page donné. Nous considérons que l'ensemble composé d'une page et des pages qui lui sont complémentaires forment une unité logique d'information. Nous avons choisi le terme ***Unité logique d'information (ULI)*** **pour désigner un document, c.-à-d. un volume d'information auto-explicative, dans le Web.**

Cette information complémentaire concernant les pages sera donc intégrée dans le fonctionnement d'un moteur de recherche. Après avoir groupé les pages dans des unités logiques d'information, nous proposons une méthode pour expliciter le contexte des pages faisant partie d'une même ULI. Le contexte des pages est donc utilisé dans l'indexation des pages dans le but améliorer la qualité de leur index. En améliorant la qualité de l'indexation d'un moteur de recherche nous nous attendons à améliorer son efficience. La découverte et l'utilisation de l'information contextuelle des pages dans le fonctionnement d'un moteur de recherche constitue la deuxième contribution de cette étude.

1.5 Organisation de la thèse

La suite de ce rapport s'organise de la manière suivante. Le deuxième chapitre est constitué d'un état de l'art sur les applications de l'analyse de liens dans le Web dont l'objectif est fournir aux utilisateur des outils d'aide pour mieux exécuter la tâche de récupération d'information. Nous nous attardons notamment sur l'application de la découverte d'unités logiques d'information pour des raisons évidentes. Nous détaillons la problématique et passons en revue les différentes méthodes proposées jusqu'à ce jour.

Dans le troisième chapitre, nous décrivons le modèle que nous proposons pour un système de recherche d'information, ou moteur de recherche, adapté à un site Web. Dans un premier

temps nous présentons une méthode pour mesurer la complémentarité entre les pages d'un site, les regrouper dans des unités logiques d'information et finalement identifier le contexte des pages faisant partie d'une même ULI. Ensuite nous suggérons une manière d'intégrer l'information contextuelle des pages dans le fonctionnement d'un moteur de recherche. Dans ce chapitre nous proposons aussi une deuxième méthode pour l'identification des contextes des pages. Il s'agit d'une extension de la première dans la mesure où plusieurs niveaux de contextes sont générés pour une page donnée au lieu d'un seul comme c'est le cas dans la première méthode. Bien entendu nous proposons aussi une manière d'intégrer la multiplicité des niveaux de contextes des pages dans le fonctionnement d'un moteur de recherche. Le modèle de moteur de recherche que nous proposons dans ce chapitre bénéficie aussi d'un langage de requêtes et d'une fonction d'appariement particuliers. Ils y sont également décrits.

Le quatrième chapitre est entièrement dédié à la partie expérimentale de notre étude. Nous y présentons d'abord la collection de tests que nous avons choisie pour tester le prototype d'un moteur de recherche implémenté selon le modèle décrit dans le troisième chapitre. Ensuite nous expliquons les expérimentations que nous avons menées, présentons et commentons les résultats. Nous y comparons notamment l'efficience de notre prototype avec celle d'un moteur de recherche traditionnel qui ne bénéficie pas de l'information contextuelle des pages.

Finalement, dans le cinquième chapitre, nous synthétisons les apports de ce travail et en tirons les conclusions. Nous citons aussi les problèmes qui restent ouverts et indiquons des pistes dont l'exploitation pourrait mener à résoudre une partie des problèmes cités.

Chapitre 2

L'analyse de liens dans la recherche d'information dans le Web

Le thème de ce chapitre est l'analyse des liens sur le Web. Nous y passons en revue des applications récentes de l'analyse des liens dans le contexte de la récupération d'information sur le Web et nous concluons avec l'application qui nous intéresse, à savoir la découverte d'unités logiques d'information dans le Web.

Depuis 1996 l'analyse de liens est de plus en plus utilisée dans le contexte d'applications dont l'objectif principal est de fournir à l'utilisateur des outils d'aide à l'exploitation de l'information disponible sur le Web. Ces applications modélisent le Web (ou une partie) par un graphe $G = (S, A)$ où S est l'ensemble des nœuds et A est l'ensemble des arêtes ou des arcs[1]. Dans ce graphe, communément appelé *graphe de citations*, chaque nœud représente une page et chaque arc (ou arête) représente un lien entre deux pages. Traditionnellement, il existe au plus un arc (une arête) entre deux nœuds, autrement dit, qu'il y ait plusieurs liens entre deux pages ou un seul lien le graphe de citations est le même. Les applications que nous citons dans la prochaine section utilisent ce graphe.

Probablement, l'application la plus commune de l'analyse des liens est celle qui concerne le classement des pages dans la liste des réponses renvoyée par un moteur de recherche. La connectivité ou la structure de liens entre les pages est analysée dans le but d'estimer la popularité des pages. Cette popularité est ensuite utilisée dans le calcul du rang final des pages dans la liste de réponses. Cependant il existe d'autres applications de l'analyse de liens moins connues que celle que nous venons d'évoquer. L'une d'entre elles est la découverte d'unités logiques d'information, l'application qui est à la base du système de recherche d'information que nous proposons dans le chapitre 3. Dans la section 2.1 nous résumons les principales applications de l'analyse de liens. Dans la section 2.2 nous introduisons l'application qui nous intéresse, la découverte d'unités logiques d'information, et nous y passons en revue les principales méthodes proposées jusqu'à ce jour et les comparons.

[1] Certaines applications modélisent le Web par un graphe non orienté, d'autres le considèrent comme un graphe orienté.

2.1 Applications principales

2.1.1 Classement de pages

Les algorithmes les plus connus de classement de pages utilisant l'analyse de liens sont l'algorithme *Page Rank* [BP98] [PBMW98] proposé par Brin et Page et l'algorithme HITS proposé par Kleinberg [Kle98].

L'hypothèse de base des deux algorithmes est qu'un lien qui mène d'une page p_i à une page p_j signifie une recommandation ou approbation de la page p_j par l'auteur de la page p_i. Notons qu'il s'agit d'une hypothèse qui n'est certainement pas toujours vérifiée dans le Web vu la diversité des types de liens qui le composent. Les deux algorithmes estiment un degré de popularité ou prestige pour les pages basé sur leurs nombres de liens entrants et sortants, autrement dit, la popularité d'une page est fonction de sa connectivité avec les autres pages. Une fois les degrés de popularité estimés, l'algorithme de classement des pages les utilisent afin de calculer le rang final des pages dans la liste de réponses.

L'algorithme *Page Rank* est basé sur la mesure éponyme, i.e. *PageRank*, qui permet d'estimer le degré de popularité d'une page. L'estimation est basée sur trois hypothèses : (i) la popularité d'une page, disons p_x, est une fonction de la popularité des pages qui la citent — les pages qui ont un lien vers p_x ; (ii) toutes les citations d'une page n'ont pas la même importance ; (iii) la popularité d'une page est indépendante des requêtes. Le fonctionnement de l'algorithme peut être résumé en deux étapes : d'abord, par rapport à une requête donnée, une fonction de correspondance sélectionne des pages selon leur contenu et le texte de l'ancre des liens qui les citent ; ensuite les pages sont triées en ordre décroissant selon l'estimation précalculée de leur popularité. L'hypothèse sous-jacente est que les pages les plus populaires sont les plus pertinentes. Cet algorithme est utilisé par le moteur de recherche `www.google.com`.

Comme nous l'avons dit plus haut, pour *PageRank* la popularité d'une page est considérée comme étant une caractéristique intrinsèque de la page dans la mesure où elle ne dépend pas des thèmes, liés à la requête, pour lesquels la fonction de classement veut estimer la pertinence de la page. Autrement dit, la notion de popularité n'est pas associée à une thématique.

Inversement, l'algorithme HITS tient compte des requête dans l'analyse de la popularité des pages. On peut considérer que dans HITS une page est analysée dans un certain contexte. Par rapport à une requête donnée, ce contexte est le sous-ensemble des pages renvoyé par un moteur de recherche traditionnel comme *Altavista* ou *eXcite* en réponse à la requête. L'analyse de liens de HITS se fait dans le graphe sous-jacent à un ensemble de pages qui est la réunion du sous-ensemble de pages que nous venons d'évoquer et des pages qui pointent sur ou sont pointées par au moins l'une des pages du sous-ensemble.

L'idée principale de HITS est que les pages dans le Web peuvent être classées comme *page autorité* ou *page index* dans un thème donné. Les pages autorités sont celles qui contiennent de l'information de bonne réputation dans un thème donné. Les pages index sont celles qui contiennent un nombre important des liens vers des pages de type autorité. En plus, Kleinberg suggère l'existence d'une relation mutuelle de renforcement entre les pages des deux

types : une bonne page de type index pointe sur plusieurs bonnes pages de type autorité et une bonne page de type autorité est pointée par plusieurs bonnes pages de type index.

La *circularité* de la relation est rompue par l'algorithme en associant à chaque page un score d'autorité, $a(p)$, et un score d'index, $h(p)$. L'algorithme calcule les deux scores de chaque page par l'itération de deux étapes : (i) le score d'index d'une page p, $h(p)$, est calculé comme étant la somme des scores d'autorité des pages pointés par la page p et (ii) d'une manière analogue le score d'autorité d'une page p est calculé comme étant la somme des scores d'index des pages qui pointent sur p. Il a été démontré que par cet algorithme les scores d'autorité et d'index des pages convergent. Expérimentalement on a montré que le nombre d'itérations nécessaire est relativement petit par rapport au nombre de pages (5 ou 6 itérations en moyenne pour un ensemble de 300 pages). Les scores finaux d'autorité (resp. d'index) des pages correspondent aux valeurs composant le vecteur propre principal du produit de matrices M^tM (resp. MM^t) où M est la matrice de connectivité associée au graphe sous-jacent à l'ensemble des pages analysées. L'élément $M(n_i, n_j)$ vaut 1 si et seulement s'il existe au moins un lien partant de la page associée au nœud n_i vers la page associée au nœud n_j, sinon $M(n_i, n_j)$ vaut 0.

Finalement les pages sont classées en ordre décroissant selon leurs scores d'index et d'autorité : deux listes sont donc proposées en sortie par le système implémentant HITS. Comme la popularité calculée par *Page Rank*, l'autorité d'une page peut aussi être associé au prestige dont une page peut bénéficier. Dans [Sou98] HITS a été utilise dans le contexte d'un moteur de recherche spécialisé dans les requêtes générales comme *Le tourisme en France* ou *La mondialisation*. Par ailleurs, HITS a été aussi exploité dans plusieurs applications dont certaines sont citées dans la suite. Contrairement à l'algorithme *Page Rank*, à notre connaissance jusqu'à ce jour aucun moteur de recherche disponible pour un usage public dans le Web n'a incorporé HITS dans son fonctionnement.

2.1.2 Collecte de pages

L'analyse de liens a aussi été envisagée pour la tâche de collecte des pages effectuée par un robot associé à un moteur de recherche. Plus précisément, l'idée est d'utiliser l'analyse de liens pour identifier des pages de bonne qualité pour l'indexation. L'hypothèse considérée ici est similaire à celle faite par *HITS* et *Page Rank* : la qualité d'une page est corrélée à sa connectivité avec les autres pages.

Dans la pratique, les robots associés aux moteurs de recherche ne collectent pas toutes les pages du Web visible pour des contraintes de temps ou d'espace. Chaque robot est soumis à une politique de parcours du graphe sous-jacent au Web. Quelle que soit sa politique de parcours, le robot dispose toujours d'un ensemble d'identifiants de pages non parcourues, notons-le P_{inexp}. Après avoir chargé une page, le robot analyse syntaxiquement le contenu de la page, récupère les identifiants des pages encore non parcourues dans les balises `<a>` et insère ces identifiants dans P_{inexp}. Une fois l'analyse d'une page finie, le robot choisit et ôte un identifiant de P_{inexp} et réitère le processus que nous venons de décrire. Considérons aussi que P_{exp} est l'ensemble des pages déjà chargées.

Au lieu de récupérer au hasard un identifiant de l'ensemble P_{inexp} et de charger la page correspondante, l'idée ici est de récupérer d'abord des identifiants des pages censées être de

bonne qualité. Pour ce faire, il faut estimer la qualité des pages identifiées dans P_{inexp}. Une solution simple pour cela consiste à récupérer d'abord les pages qui sont pointées par le plus de pages déjà chargées, c.-à-d. les pages de P_{exp}. Une autre possibilité consiste à utiliser l'algorithme *PageRank* pour calculer la popularité des pages de P_{inexp} selon leur connectivité dans le graphe sous-jacent aux pages de $P_{exp} \cup P_{inexp}$ et de considérer que la popularité d'une page comme une estimation de sa qualité. Cette approche est à la base de l'une des méthodes proposées pour le ramassage de pages testées dans [CGMP98].

L'analyse de liens a été également suggérée dans les travaux concernant les collectes focalisées. Une collecte focalisée vise à recueillir des pages traitant un ensemble préspécifié de thèmes. Dans [CvdBD99], un algorithme dont l'entrée est un ensemble de *pages exemples* (dont les thèmes incluent ceux sur lesquels la collecte est focalisée) analyse leur voisinage (en termes de la structure du graphe qui leur est sous-jacent) afin d'identifier les liens partant d'eux qui sont susceptibles de mener à des pages concernées par le(s) thème(s) de la collecte. De cette façon le robot vise à éviter de collecter des pages non concernées par les thèmes sur lesquels la collecte est focalisée.

2.1.3 La catégorisation de pages

L'analyse de liens a aussi été employée dans des méthodes visant la catégorisation de pages. La catégorisation peut être supervisée ou non supervisée. Nous appelons la catégorisation non supervisé *clusterisation* et la catégorisation supervisée *classification* (cf. section 3.2.2). Dans [PPR96], la structure des liens est utilisée pour classifier les pages dans huit classes fonctionnelles prédéfinies : page d'index, page d'accueil, etc. L'hypothèse est que la connaissance des catégories associées aux pages peut aider le processus d'exploitation d'information par les utilisateurs.

Des travaux récents ont appliqué des techniques de catégorisation pour classifier automatiquement des pages dans des répertoires thématiques, e.g. [CDI98], [Sou98], [Cha97] et [OML00]. Il s'agit donc de la construction de classifieurs automatiques. La motivation de ces travaux est que les répertoires thématiques utilisant une classification manuelle des sites (e.g. *Yahoo!*) ne peuvent pas suivre la croissance de Web. Un répertoire thématique se compose d'un ensemble de classes (qui peuvent être organisées d'une manière plate ou hiérarchique) chacune associée à un thème plus ou moins précis. Un classifieur supervisé est construit de la manière suivante : au départ chaque page d'un sous-ensemble de pages prédéfini est manuellement assigné à une ou plusieurs classes. Ensuite un algorithme d'apprentissage est utilisé afin d'apprendre au classifieur comment classer les pages en fonction des propriétés des pages de l'ensemble de départ et des classes auxquelles ces dernières ont été manuellement associées. Une fois la période d'apprentissage finie le classifieur est censé être capable d'associer automatiquement les nouvelles pages à(aux) la classe(s) qui leur est(sont) le(s) plus pertinente(s).

Traditionnellement, les classifieurs se basent sur le contenu des pages pour les assigner aux différentes classes. L'idée ici est de se servir des indicateurs additionnels contenus dans les hypertextes, à savoir les liens entre les pages, pour mieux les classifier. Illustrons ici deux travaux adoptant une tel type d'approche.

Dans [CDI98] le voisinage d'une page donnée, disons p_i, est pris en considération dans sa

classification. En classifiant p_i l'algorithme analyse les pages voisines, celles qui pointent ou sont pointées par p_i. Les classes des pages de ce voisinage (si elles ont déjà été classées) et leur contenu textuel sont employées pour déterminer la classe de p_i. Deux situations sont possibles lors de la classification de p_i selon que toutes ses pages voisines ont déjà été classifiées, ou juste un sous-ensemble d'elles l'a été. Dans les deux situations l'algorithme proposé dans [CDI98] a réussi à augmenter l'efficacité du classifieur. Dans une étude plus récente [OML00] Oh et al. suggèrent un nouvel algorithme, semblable à celui proposé dans [CDI98] mais intégrant deux innovations : (i) les contenus des pages voisines d'une page en train d'être classée, disons p_i, n'influencent pas équitablement la définition de la classe de p_i et (ii) les classes des pages voisines de p_i n'influencent pas équitablement la définition de la classe de p_i. Ces deux innovations sont liées à un taux de fidélité calculé pour chaque lien attachant p_i à une autre page. Cette dernière méthode a montré une amélioration significative d'efficacité par rapport à celle atteinte par [CDI98].

2.1.4 La découverte de pages similaires

Un genre intéressant de recherche est la *recherche par exemples*. À la différence de la recherche traditionnelle où l'on formule une requête composée d'une liste de mots dans le but de récupérer des documents qui lui sont pertinents, dans la recherche par exemples l'utilisateur indique au système un document pertinent à son besoin d'information et le système est censé retrouver des documents similaires thématiquement au document exemple fourni. La fonction *What's Related* présente dans le navigateur *Netscape* est une implémentation d'une recherche par exemples. Une implémentation simple de ce type de recherche consiste à grouper les pages selon leur contenu à l'aide une méthode de clusterisation, les pages similaires étant regroupées dans des clusters. Cependant, cela n'est pas envisageable dans le contexte du Web vu la taille de ce dernier en termes de nombre de documents. Le système *Hypursuit* décrit dans [Ron96] à proposé une clusterisation de pages similaires basé non seulement sur le contenu des pages mais aussi sur la structure les reliant.

Dans [DH99] deux méthodes exploitant l'analyse de liens sont suggérées pour l'identification des pages similaires. La première est basée sur l'algorithme HITS de Kleinberg évoqué dans la section 2.1.1. Prenons comme exemple une requête spécifiant l'URL d'une page p_i. Pour construire l'ensemble initial de pages requis par HITS, les ensembles suivants de pages sont réunis : les parents de p_i (pages qui pointent sur p_i), les enfants de ses parents (pages pointées par les parents de p_i), les enfants de p_i et les parents de ses enfants. HITS est alors appliqué à cet ensemble et les pages assignées aux scores d'autorité les plus élevées sont considérées comme celles étant les plus similaires à p_i.

La deuxième méthode proposée dans [DH99] s'inspire de la méthode des co-citations [Sma73] issue de la bibliométrie. L'idée est d'identifier les pages qui pointent sur la page indiquée dans la requête (p_i) et de découvrir vers qui d'autre ces pages pointent. Les pages qui sont souvent co-citées avec la page indiquée dans la requête sont considérées comme celles étant les plus similaires à p_i. Bien que ce soit une méthode assez simple, elle s'est avérée aussi performante que la première.

2.1.5 La découverte de sites dupliqués

Une autre application de l'analyse de liens sur le Web concerne la localisation d'information dupliquée. Identifier ce type d'information peut être utile dans plusieurs contextes. Par exemple, des copies non-autorisées de pages ou de sites entiers pourraient être découverts, de même que des pages bien répandues dans le Web comme les *FAQs*[2] pourraient être localisées lors de mises à jours. Dans le contexte des moteurs de recherche, la connaissance de pages ou sites dupliqués peut être utilisée (i) dans le but d'éviter qu'un même contenu soit indexé plusieurs fois et (ii) afin de ne pas rendre dans la liste de réponses des pages de contenu quasi identiques. La deuxième utilité est particulièrement intéressante pour les métamoteurs.

Certaines méthodes basées seulement sur le contenu des pages ont été suggérées. Par exemple, la méthode décrite dans [BKM+00] propose la création d'un résumé pour chaque page basé sur son contenu. Ensuite une mesure de ressemblance entre deux pages est définie. Cette mesure est particulièrement utile dans l'identification de pages qui sont quasi identiques (les différences seraient due à des changements d'en-tête, à l'addition/suppression/édition de liens, à différentes signatures en bas de page, etc.). Notons que la mesure de similarité dont il est question dans [BKM+00] est différente des mesures de similarité sémantique traditionnellement utilisées par les systèmes de recherche d'information. Des techniques similaires à celles décrites dans [BKM+00] ont été proposées dans [BB99] pour calculer la ressemblance entre sites afin de localiser des sites dupliqués.

D'autres études plus récentes suggèrent l'utilisation de l'analyse de liens pour localiser des ressources Web (pages ou sites) dupliquées. Dans [BBDH00] une comparaison de différentes méthodes de localisation de sites dupliqués est proposée. Des méthodes comparées utilisent plusieurs types d'analyse : l'analyse d'adresses IP, l'analyse de la structure de répertoire codée dans l'URL des pages, l'analyse des liens entre les pages, etc. Pour ce qui est de l'analyse des liens, l'idée de base est la suivante : deux sites sont dupliqués si leurs pages sont associées à des URLs similaires et si les pages ayant des URLs similaires partagent un nombre important de liens sortants.

Par ailleurs, dans [BBDH00] Bharat signale les conséquences que la duplication d'information peut avoir sur les moteurs de recherche dont les algorithmes utilisent la structure du graphe sous-jacent aux pages indexées. Une perturbation de cette structure peut affecter considérablement l'efficacité des algorithmes.

2.1.6 La découverte de la portée géographique de ressources Web

Quand une ressource (page ou site) Web est créée son(es) auteur(s) cible(nt) un certain type de lecteur ou d'audience. Les audiences peuvent cependant évoluer, atteignant des types de lecteurs pas considérés au départ par l'auteur. La portée géographique d'une ressource Web est le secteur géographique où son audience est localisée à un moment donné. Par exemple, un site Web contenant de l'information sur les locations d'appartement ou les restaurants concerne principalement la communauté géographiquement située dans la proximité de ces endroits. En revanche, nous pouvons imaginer des ressources Web avec des portées associées à des communautés géographiques plus étendues. Par exemple, la portée du site du journal

[2]En anglais *Frequently Asked Questions pages.*

Le Monde est beaucoup plus large que celle du site de l'agenda stéphanois `www.lagenda.net` qui intéresse principalement les résidents de la région de Saint-Etienne.

La portée géographique d'une information peut être utile à la navigation mais aussi à la recherche d'information à l'aide d'un moteur de recherche. Par exemple, si un moteur de recherche dispose de la portée géographique des données indexées, lors d'une requête, il peut essayer d'identifier géographiquement d'où la requête est issue (e.g. à travers un profil prédéfini de l'utilisateur, ou l'adresse IP de la machine ayant soumis la requête). Puis en fonction de la portée géographique le moteur peut soit filtrer les résultats (e.g. montrer juste les ressources ayant la même portée géographique que celle de la requête) ou distinguer visuellement dans les réponses les ressources locales (par rapport à la requête) des ressources non locales. L'utilité de la portée géographique des données est plus claire dans la navigation. Par exemple certains répertoires thématiques comme *Yahoo!* ou *Looksmart* proposent des catégories associées à des portées géographiques spécifiques (e.g. la catégorie concernant l'état américain de Hawaii est disponible sur `http://dir.yahoo.com/Regional/U_S__States/Hawaii/`). Dans ces répertoires les ressources sont classées manuellement. Les catégories sont censées aider l'utilisateur dans sa navigation.

Buyukkokten et al. [BCGM+99] et Ding et al. [DGS00] suggèrent une méthode pour découvrir automatiquement la portée géographique de ressources Web. Elle est basée sur l'origine géographique des pages ayant des liens qui pointent vers la ressource analysée, disons R^*. Une ressource R_x est considérée être dans la portée géographique de R^* si (i) un pourcentage élevé de pages de R_x pointent vers R^* et (ii) si les pages qui pointent vers R^* sont relativement distribuées dans R_x. Les expérimentations menées ont montré que cette méthode est aussi efficace qu'une méthode alternative basée sur le contenu des pages, également décrite dans [DGS00]. La méthode basée sur les liens a cependant l'avantage d'être plus efficiente que la méthode basée sur le contenu.

2.1.7 La découverte de la réputation de ressources Web

Un autre application innovante de l'analyse de liens est celle proposée par Mendelzon et al. dans [MM97] et [RM00]. Fondamentalement, les moteurs de recherche reçoivent des requêtes composées d'ensemble de mots-clés correspondant à des thèmes et, en réponse à elles, ils doivent retourner des pages concernant ces thèmes. L'idée dans [RM00] est de concevoir un système fonctionnant de la manière opposée : étant donnée une page (ou un site) le système doit rechercher les thèmes qui la (le) concernent. L'idée réellement innovante est celle de ne pas envoyer tous les thèmes de la ressource mais seuls ceux pour lesquels la ressource a une bonne réputation. En effet, une ressource n'a pas nécessairement une bonne réputation pour tous les thèmes concernés par l'information qu'elle véhicule.

Considérons l'anecdote suivante pour illustrer notre dernière affirmation. Dans le but de recueillir les impressions d'un groupe de personnes sur son site récemment remodelé, une université a organisé une enquête. Parmi les questions de cette enquête, une portait sur la réputation du site, i.e. la raison principale pour laquelle les personnes accédaient au site. Étrangement, les résultats de l'enquête ont révélé que la réputation du site était lié à une bibliothèque de fichiers d'images animées. Les personnes enquêtées accédaient au site pour récupérer ces petites images pour les inclure dans leurs pages personnelles.

Il existe plusieurs applications pour une telle analyse. Certaines organisations dépensent de grosses sommes d'argent dans des études dont le but est de savoir comment leurs sites sont perçus par le public. L'analyse automatique d'une ressource Web peut révéler les thèmes où sa réputation est anormalement élevée ou mauvaise. La section précédente concernait les portées géographiques des ressources. La réputation d'une ressource Web peut être considérée comme un genre particulier de portée d'une ressource. Au lieu de géographique, la réputation d'une ressource peut être considérée comme sa portée thématique.

Pour ce qui est des méthodes proposées pour la découverte automatique de la réputation de ressources Web la seule que nous connaissons est celle issue des études de Mendelzon [MM97] et [RM00]. Dans cette méthode les thèmes des pages qui pointent vers la ressource dont on veut connaître la réputation sont analysés dans le but d'identifier une première liste de thèmes. Ces thèmes sont ceux pour lesquels la ressource analysée pourrait éventuellement avoir une bonne réputation. Une fois cette liste de thèmes établie, le système estime la réputation de la ressource dans chacun des thèmes à travers une extension de l'algorithme *Page Rank*, évoqué dans la section 2.1.1. Dans sa version originale *Page Rank* calcule pour chaque page un degré de prestige qui est basé sur la popularité de la page en termes de sa connectivé dans le graphe. Ce prestige est une qualité intrinsèque de la page, il n'est pas associé à un thème donné. Dans [RM00] on modifie cet algorithme de sorte que la popularité estimée pour chaque page soit dépendante d'un topique donné. Un prototype de ce système est disponible sur `http://www.cs.toronto.edu/db/topic/`. Notons que l'utilisateur peut aussi demander au système de calculer la réputation d'une page directement sur un topique donné.

2.1.8 La découverte des communautés dans le Web

Une communauté dans le Web peut être définie comme un ensemble de ressources partageant un centre d'intérêt thématique. Quelques algorithmes ont été récemment suggérés pour identifier automatiquement de telles communautés. L'application évidente de tels algorithmes est la classification automatique des ressources du Web dans les catégories de répertoires thématiques tels que *Yahoo!* ou *Infoseek*. Comme signalé dans [KRRT99], la découverte des communautés d'intérêt sur des thèmes précis peut être aussi extrêmement utile pour définir les cibles d'une campagne publicitaire.

Notre intérêt ici porte sur les approches basées sur la structure de liens. Deux approches de ce type sont proposées dans [GKR98] and [KRRT99]. L'approche suggérée dans [GKR98] est basée sur l'algorithme HITS de Kleinberg. Comme évoqué dans la section 2.1.1, par rapport à un topique (requête) donné, HITS retourne deux listes : les k-meilleures pages d'index et les k-meilleures pages d'autorité. Les scores d'index et d'autorité associés à ces pages correspondent aux valeurs composant les vecteurs propres principaux des produits de matrices M^tM et MM^t. M est la matrice de connectivité associé au graphe sous-jacent à un ensemble de pages, notons-le S_{extend} défini au début de l'exécution de l'algorithme. Dans [GKR98] l'ensemble composé des n-meilleures pages ($n < k$) d'autorité et d'index est considéré comme une communauté d'intérêt associé au(x) topique(s) spécifié(s) dans la requête qui constitue le point de départ de HITS. Cette communauté est appelé *communauté principale* par les auteurs de l'étude.

Parallèlement, on a observé l'existence de certaines pages légèrement hors-topique dans les k-meilleures pages d'index et les k-meilleures pages d'autorité renvoyées par HITS. Ces pages formaient en fait d'autres communautés plus ou moins liées à la communauté principale. L'apparition de ces pages hors-topique dans la liste initiale renvoyée par HITS est due au problème appelé *la dérive de topique* associé à l'algorithme original. Ce problème est lié à la construction du graphe sous-jacent à S_{extend} que l'algorithme exploite (cf. [GKR98]). Les communautés hors-topique générées par HITS correspondent en fait à des régions du graphe sous-jacent à S_{extend} où la densité des liens est moins élevé que celle observée dans la région du graphe qui regroupe les pages formant la communauté principale.

L'apport principal de [GKR98] est la suggestion que les communautés hors-sujet pourraient être déduites des vecteurs propres secondaires des mêmes produits de matrices utilisés pour trouver la communauté principale. Ainsi, on suggère que HITS peut trouver non seulement une communauté principale concernant une requête, mais aussi d'autres communautés plus ou moins liées à la première.

Une autre approche basée sur l'analyse de liens pour découvrir des communautés d'intérêt sur le Web est proposée dans [KRRT99]. Cette approche se focalise sur les communautés *émergentes*. Les communautés de ce type sont normalement associées à des thèmes très spécialisés. De ce fait, même si ces communautés atteignent une taille importante elles seront difficilement intégrées dans des répertoires majeurs du Web tel que *Yahoo!* ou *Open Directory*. La méthode vise à identifier des communautés émergentes par leurs *signatures*. Dans [KRRT99] on fait l'hypothèse que la signature d'une communauté peut être représentée par un graphe biparti avec une densité élevé de liens. La théorie des graphes dit qu'un tel graphe biparti possède avec une probabilité élevée un sous-graphe qui est biparti et complet. Ainsi la méthode consiste dans la recherche de sous-graphes bipartis complets dans le Web. Les expérimentations évoquées dans l'étude ont été faites sur une photographie du Web construite par un robot. On a découvert plus de 100000 signatures. L'analyse manuelle de 400 signatures tirées au hasard a détecté moins de 5% de signatures dont l'existence semblaient être une coïncidence, autrement dit, des signatures qui ne correspondaient pas à des communautés d'intérêt.

2.2 Découverte d'unités logiques d'information

Dans cette section, nous nous intéressons à la problématique de la granularité de l'information disponible dans le Web. Dans ce qui suit, nous expliquons la problématique avec un exemple pratique, ensuite nous passons en revue des différentes approches proposées jusqu'à ce jour concernant cette problématique. Afin de faciliter la comparaison des différentes approches, nous avons dégagé un ensemble de caractéristiques leur étant communes, et ensuite nous les avons classées d'après ces caractéristiques. Nous avons choisi l'une des caractéristiques pour cataloguer les approches. Finalement, nous proposons un tableau comparatif qui permet de mieux situer les approches les unes par rapport aux autres et d'introduire la nôtre.

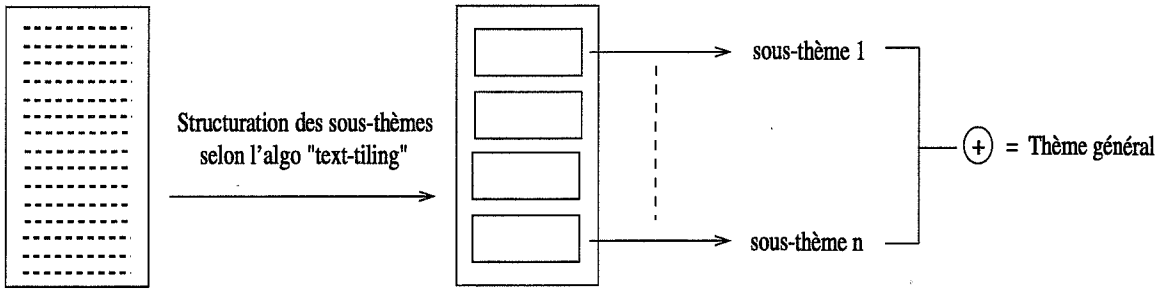

FIG. 2.1: Structuration d'un document en sous-thèmes.

2.2.1 Problématique

Dans ses travaux concernant la structuration automatique d'un document en sous-thèmes (cf. l'algorithme *TextTiling* dans [HP93]) Hearst considère un document en texte intégral[3] comme étant composé d'un thème général qui spécifie le contexte de l'information contenue dans le document et des sous-thèmes plus spécifiques qui doivent être compris dans le contexte du thème général. Le thème général peut être vu comme la toile de fond du document où viennent se greffer les sous-thèmes. Le contexte de l'étude publiée dans [HP93] est d'indexer plus précisément un document à travers de l'identification de non seulement son thème général mais aussi celle de ses sous-thèmes. La méthode proposée divise un document en une séquence de segments et, à travers une analyse de similarité du contenu des segments successifs, regroupe certains segments en sous-thèmes. Ainsi, après traitement, le document est vu comme une séquence de sous-thèmes. Le thème général est considéré comme étant le document entier, i.e. la séquence de tous les sous-thèmes découverts.

Il semble évident que la compréhension des sous-thèmes d'un document soit conditionnée à la compréhension du thème général. Autrement dit, le contexte du document est fondamental pour la compréhension des sous-thèmes du document. Nous pouvons généraliser cette idée et suggérer que la compréhension d'une *pièce d'information* quelconque est conditionnée à la compréhension du contexte de cette pièce d'information.

Lorsqu'un document est créé directement ou traduit sous forme hypertextuelle il est très souvent fractionné en plusieurs parties, chaque partie étant associée à un nœud différent du graphe sous-jacent au système hypertexte en question. Ainsi, un document, tel que nous l'entendons[4], ne se compose pas d'un objet unique dans un hypertexte mais d'un conglomérat de plusieurs objets. Tel que nous l'avons précisé dans l'introduction, dans ce rapport nous utilisons le terme *unité logique d'information* pour désigner un document hypertextuel.

En ce qui concerne la *navigation*, mécanisme de base pour la tâche de récupération d'information dans un hypertexte, le fractionnement des documents en plusieurs nœuds peut agir sur la facilité/difficulté de compréhension de l'information disponible. Cela dépend, en grande

[3]Texte intégral en opposition aux documents de type résumé ou dépêches d'actualité qui sont généralement courts. Avant l'apparition des bases de données capables de stocker et gérer des documents en texte intégral, les systèmes de recherche d'information traitaient traditionnellement des textes relativement courts.
[4]Un volume d'information auto-explicative.

partie, de la complexité de la structure de liens reliant les nœuds et de l'identification des documents composés par des groupes de nœuds. Toutefois, pour ce qui nous concerne, il est question de systèmes de recherche d'information appliqués à un système hypertexte ; analysons donc les conséquences du fractionnement d'un document en plusieurs nœuds vis-à-vis de la modélisation d'un tel système et de la tâche de recherche exécutée par lui.

Si nous associons ce fractionnement d'un document en nœuds avec ce qui a été dit plus haut sur l'importance du contexte dans la compréhension d'une information, nous pouvons dire que la compréhension par un acteur donné (humain ou logiciel) de l'information contenue dans un nœud donné n_i dépend de la connaissance que cet acteur possède des autres nœuds composant le même document que n_i. Considérons un document d_x fractionné dans un ensemble de nœuds S_x et un acteur en train de visiter le nœud $n_i \in S_x$. Si l'acteur n'a pas la connaissance de certains autres nœuds contenus dans S_x, il éprouvera des difficultés à comprendre, voire ne comprendra pas, l'information véhiculée par n_i. Nous pouvons dire que ces autres nœuds de S contextualisent ou complètent le contenu de n_i.

Pour ce qui est des systèmes de recherche d'information appliqués aux systèmes hypertextes, afin que l'information contenue dans un nœud soit correctement indexée, le processus responsable pour l'indexation doit donc se baser non seulement sur le contenu de n_i mais aussi sur le contenu des autres nœuds qui donnent le contexte de l'information véhiculée par n_i. Si l'index du nœud n_i est extrait uniquement à partir de son contenu, la qualité de l'index sera moindre.

Ainsi, la connaissance des nœuds qui forment ensemble un document est fondamentale pour une bonne indexation des nœuds. Cependant, sauf dans des systèmes hypertextes bien particuliers où les liens sont typés et les documents sont facilement identifiables, l'association d'un document avec les nœuds qui le composent n'est pas explicite, autrement dit, les frontières d'un document hypertextuel ne sont pas clairement définies. Nous pouvons faire une analogie avec le travail de structuration d'un document en sous-thèmes de Hearst et considérer que, dans un système hypertexte, il n'existe quasiment que des nœuds représentant les sous-thèmes d'un document plus conséquent[5]. La difficulté consiste donc à déterminer quels sont les nœuds qui contextualisent ou complètent l'information contenue dans un nœud donné.

La problématique est analogue à celle abordée par Hearst dans [HP93] excepté le fait qu'elle se place dans une direction opposée. Dans [HP93], à partir d'un document en texte intégral, Hearst essaie d'identifier les sous-thèmes du document. Il est question d'analyse d'un document dont on connaît les frontières en essayant de le ***diviser*** et en identifier les parties. En revanche, la problématique ici est de ***(re-)composer*** le document en mettant ensemble ses parties. Cette problématique peut être considérée comme plus complexe que celle abordée par Hearst dans la mesure où si d'une part on connaît l'ensemble des parties (les nœuds), d'autre part on ne connaît pas quel sous-ensemble de cet ensemble compose un document. Autrement dit, ce ne sont pas tous les nœuds de l'ensemble qui forment un document donné[6].

[5] Dans certaines situations, un document peut être associé à un seul nœud.

[6] Nous pouvons imaginer l'existence de documents hypertextuels contenant de l'information à des granularités d'information différentes. Ainsi, un document très générique serait celui composé par tous les nœuds de l'hypertexte considéré. Dans ce passage nous faisons référence à un document hypertexte de granularité

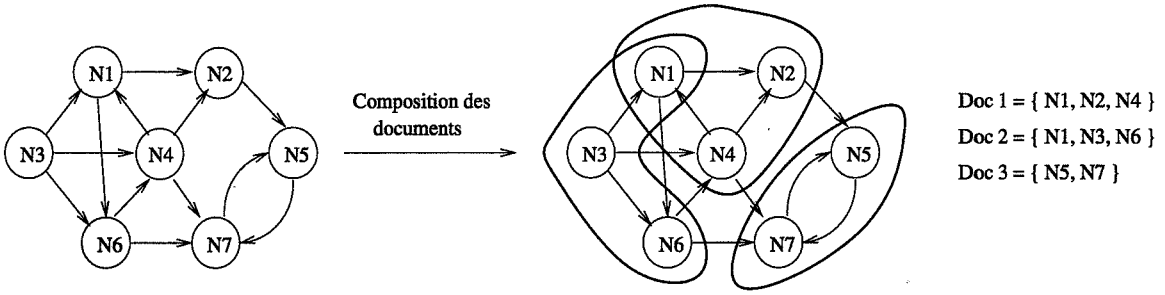

FIG. 2.2: Composition de documents à partir de nœuds.

Les nœuds de l'ensemble forment plusieurs documents à la fois. De plus, un même nœud peut éventuellement faire partie de plusieurs documents et non pas d'un seul. Une deuxième différence est que Hearst manipule des documents séquentiels alors que nous manipulons des documents hypertextuels pour lesquels il n'y a pas de parcours séquentiel préétabli de lecture.

Dans le Web, le fractionnement d'un document en plusieurs parties est encore plus remarquable que dans un hypertexte traditionnel. Si le caractère hypertextuel du Web implique naturellement le fractionnement des documents, on peut observer un style de rédaction particulier pratiqué dans le Web. En effet, de nombreux chercheurs dans le domaine de l'hypertexte et de la recherche d'information appliquée aux systèmes hypertextes suggèrent que l'hypertexte doit être considéré comme étant un nouveau genre linguistique dont les particularités devraient être prises en compte lors du développement de systèmes manipulant de l'information codée de cette façon. Des règles d'édition latentes telles que « communication maximale avec effort minimal » [Ami97] déduites par l'analyse de la présentation de données disponibles dans le Web suggèrent l'édition de pages qui ne dépassent pas une certaine longueur, e.g. une ou deux fois la hauteur d'un écran traditionnel de 17 pouces.

Quelles sont les implications du fractionnement des documents au niveau de la structure et du contenu d'un ensemble de pages créées par un auteur? Aussi bien dans un système hypertexte que dans le Web, l'auteur définit le contenu des nœuds/pages et la structure entre eux en supposant vraie une hypothèse de base : lorsqu'un acteur donné (humain ou logiciel) commence à lire une page, disons p_i, il a déjà parcouru un chemin de pages avant d'arriver à p_i[7]. Les pages se trouvant dans le chemin lui auront apporté suffisamment d'informations pour qu'il comprenne le contenu de p_i. Autrement dit, les pages parcourues au préalable lui auront donné le contexte de l'information de la page courante. La considération de cette hypothèse par l'auteur influence fortement son style de rédaction, c.-à.-d. aussi bien le contenu des pages que la structure qui les relie. Par exemple, pour deux pages p_x et p_y où p_x donne le contexte de p_y, le lecteur est censé être passé par p_x avant d'arriver à p_y. Si d'un côté il est probable que p_x et p_y soient reliées par un chemin qui est probablement court, de l'autre côté,

minimale.

[7]Il existe trois manières générales d'accéder à une page : (i) en saisissant directement une URL connue dans le navigateur, (ii) à travers un moteur de recherche ou un annuaire, ou (iii) à travers la navigation (sans passer par un moteur ou un annuaire). L'auteur des pages les crée en supposant que le lecteur a utilisé le dernier moyen d'accès pour atteindre la page.

malgré le fait que le contenu de p_x soit nécessaire pour la compréhension de l'information vehiculée p_y, ce contenu (ou une partie) ne se retrouvera pas en double dans p_y.

Nous constatons aussi dans le Web ce que nous avons observé dans un système hypertexte traditionnel, à savoir que le contenu d'une page Web n'exprime pas toute l'information qu'elle véhicule. Pour qu'un acteur puisse comprendre l'information véhiculée par une page, il lui faut connaître aussi le contenu des pages qui complètent ou contextualisent la page en train d'être lue. D'une manière analogue à l'observation faite plus haut, une page doit être lue dans le contexte du document dont elle fait partie et non pas individuellement. Considérons alors un ensemble de pages composant un document devant être indexées par un moteur de recherche. Chacune des pages de ce document doit être indexée non seulement à partir du contenu de la page elle-même mais aussi du contenu des pages qui forment le même document qu'elle et la contextualisent. Malheureusement, la situation est similaire à celle rencontrée dans des systèmes hypertextes traditionnels dans la mesure où les frontières d'un document ne sont pas explicites. La difficulté consiste donc à définir ces frontières, autrement dit, définir pour une page l'ensemble de pages qui la contextualisent/complètent.

Une manière de rendre explicite les frontières des documents disponibles dans le Web serait d'utiliser des liens typés du genre *est-partie-de* où *est-composé-de*. En effet, depuis la version 2.0 de la DTD qui définit le HTML, il existe pour l'élément `<A>` les attributs `rel` et `rev` qui servent à rendre explicite la relation existante entre la page pointante — celle d'où le lien part — et la page pointée. Cependant, ces attributs ne sont utilisés que dans très peu de pages. De plus, il nous semble utopique que des valeurs prédéfinies pour ces attributs puissent convenir à une communauté aussi large que celle du Web, ne serait-ce que par le fait que certains sites pourraient avoir des besoins spécifiques pour ce qui concerne les valeurs de ces attributs. Il serait préférable de développer un langage de définition de types de liens et non pas de fixer les types des liens eux-mêmes. Ainsi, si le problème de la syntaxe est résolu, celui de la sémantique reste ouvert. Un accord sur la sémantique ne peut être trouvé que dans des configurations particulières comme par exemple au sein d'un nombre restreint de sites ayant des thématiques communes. Mais ceci est une discussion qui ne rentre pas dans le thème que nous abordons dans ce rapport.

Ce n'est donc pas par le typage des liens que les pages composant un même document peuvent être identifiées. Il est important de signaler que, dans la plupart des cas, un être humain est capable d'identifier les pages composant une même unité logique d'information, i.e. les pages composant un même document. Comme les liens ne sont pas typés dans le Web, le lecteur se sert de quelques indicateurs pour naviguer au sein d'un même document. Par exemple, la mise en forme des ancres des liens, leur texte ou leur emplacement dans la page permettent très souvent à l'utilisateur de savoir si la page pointée par un lien fait ou non partie du même document que la page courante. Imaginons qu'un lecteur soit en train de lire une page sur les thématiques principales d'un projet européen de recherche et, qu'en bas de cette page, il y ait trois liens. Supposons que les textes des ancres de deux des trois liens soient *Détails de la première thématique* et *Détails de la deuxième thématique* et que le texte de l'ancre du troisième lien soit *Sponsors du projet*. Si le focus de l'utilisateur est sur les thématiques du projet il veut continuer sa lecture dans le document concernant les thématiques du projet, et ainsi, il est évident qu'il ne suivra pas le lien *Sponsors du projet*. Le texte de

l'ancre du dernier lien indique clairement que la page renvoyée par le lien ne concerne pas les thématiques du projet, alors que les textes des deux autres liens donnent des indications claires[8] que les pages pointées font aussi partie du document concernant les thématiques du projet.

La capacité de déduction d'un lecteur citée dans le dernier exemple n'existe pas dans les moteurs de recherche traditionnels. Les moteurs *grand-public* disponibles à l'heure actuelle ignorent le fait que lorsqu'un document est représenté sous forme hypertextuelle il est fractionné en plusieurs pages. Les raisons de cette attitude seront abordées tout au long de ce chapitre. D'une manière générale, nous pouvons avancer que la découverte des ULIs demande l'application d'une méthode qui peut s'avérer très coûteuse en termes de temps de calcul. L'augmentation du temps de réponse du moteur de recherche éventuellement impliqué par la découverte/manipulation de documents et non pas de pages est un facteur auquel les utilisateurs sont très sensibles [KT00]. Dans la suite, nous allons illustrer à l'aide d'un exemple simple ce que le fait d'ignorer le fractionnement d'un document en plusieurs nœuds implique au niveau du fonctionnement et de l'efficacité d'un moteur de recherche. Ensuite, nous aborderons comment les moteurs de recherche pourraient prendre en compte le fractionnement des documents.

Prenons par exemple un utilisateur qui veut trouver de l'information concernant des notes de cours sur les graphes dans le contexte d'un cours de structure de données. Dans le pire des cas, l'utilisateur saisira une requête contenant un seul terme, e.g. *structure de données*, ou *graphes*, ou, pire encore, *cours*. Dans le meilleur des cas, il saisira une requête contenant au moins les trois termes principaux qui constituent son besoin d'information, i.e. *structure de données* AND *graphes* AND *cours*[9]. Cependant, comme illustré sur la figure 2.3, il se trouve que souvent les ensembles de pages contenant de l'information sur l'enseignement dans une université sont construits de sorte qu'il y ait une page pour tous les cours proposés de laquelle partent des liens vers des pages individuelles à chaque cours. Ensuite, une page concernant un cours donné pointe sur une page contenant une liste de pointeurs vers des pages contenant chacune les notes de cours d'une séance donnée. À ce dernier niveau, nous rencontrons finalement une page répondant au besoin d'information exprimé par la requête, i.e. des notes de cours de théorie de graphes dans le contexte d'un cours de structure de données.

Il est peu probable que la page concernant les notes de cours sur la théorie des graphes contienne les trois termes cités plus haut : *cours*, *graphes* et *structure de données*. Par conséquent, la page pertinente au besoin d'information de l'utilisateur, en l'occurrence P_8, n'est pas retournée à l'utilisateur en réponse à la requête *structure de données* AND *graphes* AND *cours*[10]. Par ailleurs, à une requête contenant deux des trois termes, e.g. *graphes* AND *cours*, ou à une autre contenant uniquement l'un des trois termes, e.g. *graphes*[11], un moteur de recherche traditionnel renvoie beaucoup trop de réponses (pour la plupart non-pertinentes) qui finissent par noyer le sous-ensemble de réponses pertinentes.

[8] Même si elles peuvent ne pas se vérifier.

[9] En supposant que le langage de requête utilisé par le moteur de recherche accepte les requêtes booléenes

[10] La requête *structure de données* AND *graphes* AND *cours* lancée le 25/05/2001 sur *Altavista* a reçu 267199 réponses.

[11] La requête *graphe* lancée 25/05/2001 sur *Altavista* a reçu 410292 réponses.

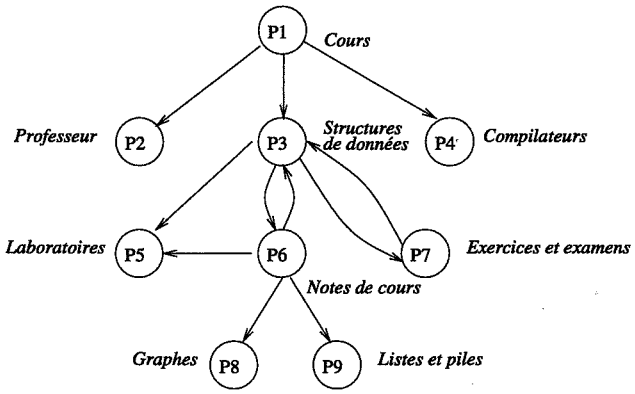

FIG. 2.3: Partie du graphe du Web illustrant la région où l'une des notes de cours sur les graphes serait disponible.

Actuellement, pour résoudre ce problème, un utilisateur procède en deux étapes. Dans la première étape il essaye d'isoler les pages sur les cours de structure de données en soumettant la requête *structure de données* AND *cours*. Ensuite, il analyse manuellement les pages retournées afin d'identifier celles qui mènent à des notes de cours concernant les graphes. Évidemment, l'inconvénient de cette technique est sa deuxième partie qui doit être accomplie manuellement.

Quelle est la raison pour laquelle la page pertinente, i.e. p_8 (cf. figure 2.3), n'a pas été retournée comme réponse à la requête *structure de données* AND *graphes* AND *cours* ? Simplement parce que, lors de l'indexation, le moteur de recherche l'a considérée comme étant un document à part entière et, par conséquent, son index a été basé uniquement sur les termes de son contenu. Puisque les termes *cours* et *structure de données* ne se trouvent pas dans le contenu de p_8, l'index de p_8 ne peut pas contenir ces termes et, par conséquent, la page n'est pas jugée pertinente à la question posée. Pour que p_8 ait été considérée comme pertinente, il aurait fallu que les trois termes de la requête aient été inclus dans son index. Et pour cela, au moment de la création de l'index de p_8, le système aurait dû se rendre compte que le contenu p_8 était fortement lié à celui des pages p_6 et p_3 et que, par conséquent, l'index de p_8 aurait du être basé non seulement sur son contenu mais aussi sur ceux de p_6 et p_3. Autrement dit, il aurait fallu que le module d'indexation se rende compte que p_8, p_6 et p_3 formaient un document ou un ensemble logique d'information et que les informations véhiculées par elles étaient fortement liées.

Plus haut nous avons observé que certains indicateurs sur les pages permettaient à un lecteur humain de distinguer les frontières d'un document. Ainsi, s'il était question d'une indexation manuelle dans l'exemple précédent, p_8 serait probablement correctement indexée puisque l'auteur de l'indexation se serait rendu compte que p_6 et p_3 complètent le contenu de p_8. Par contre, nous ne sommes pas intéressés aux modules d'indexation manuelle[12]. Les

[12] Un module d'indexation manuelle peut techniquement être envisagé dans un moteur de recherche mais, dans la pratique, c'est une démarche impossible dès que le volume de pages à indexer devient important.

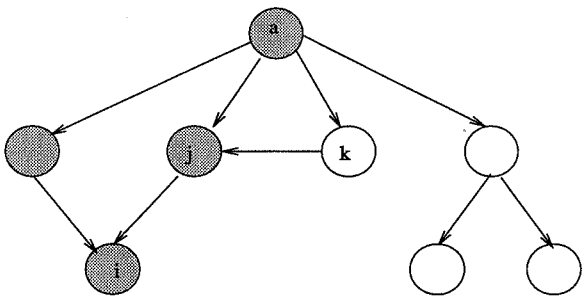

FIG. 2.4: Parcours en profondeur d'un graphe (les nœuds déjà visités sont grisés). Le nœud en train d'être visité est le nœud k.

annuaires tels que *Yahoo!* (`www.yahoo.com`) et *The Open Directory Project* (`www.dmoz.org`) utilisent en effet une indexation manuelle, mais il est important de signaler que le volume de données indexé par les répertoires thématiques est beaucoup plus petite que celle traitée par les moteurs de recherche. En effet, contrairement aux moteurs de recherche, les répertoires thématiques ne se veulent pas exhaustifs, et de plus ils indexent des sites et non pas des pages. De ce fait ce genre d'outil a un champ de travail beaucoup moins étendu que celui des moteurs de recherche. La conclusion est que l'indexation automatique est une caractéristique indissociable des moteurs de recherche.

La question logique qui suit concerne la façon dont le module d'indexation pourrait prendre en compte la fractionnement des documents lors de l'indexation des pages. On pourrait imaginer qu'un acteur logiciel, en l'occurrence le robot d'un moteur de recherche, soit capable d'identifier les frontières des documents pendant son parcours[13] de la même façon qu'un acteur humain l'est. Pour ce faire on pourrait envisager des heuristiques qui permettraient au robot de déterminer si deux pages liées par un lien sont complémentaires ou non et cela en s'appuyant sur les indicateurs évoqués plus haut. Bien qu'envisageable, une telle méthode serait difficile à concevoir puisque la déduction de complémentarité entre les pages serait faite pendant le parcours du robot et, de ce fait, elle se baserait sur un graphe incomplet. En effet, l'intuition dit qu'une analyse du graphe complet est plus riche en information pour l'analyse de complémentarité qu'une analyse effectuée sur un graphe incomplet. L'autre inconvénient de la découverte des documents pendant le parcours du robot consiste dans le fait que les algorithmes de parcours de graphes misent généralement sur l'efficacité, par exemple, en évitant de visiter deux fois un même nœud. Ainsi, les liens qui mènent vers des pages déjà visitées ne sont pas traversés et, par conséquent, certains chemins menant à une page donnée peuvent ne pas être parcourus/identifiés pendant le parcours. Par exemple, pendant un parcours en profondeur exécuté sur le graphe illustré dans la figure 2.4, le chemin $a \rightarrow k \rightarrow j \rightarrow i$ n'est pas parcouru par le robot puisque le nœud j a déjà été visité lorsque l'on arrive au nœud k.

En effet, un algorithme de parcours de graphe traditionnel peut être modifié de sorte que tous les chemins menant aux différentes pages soit parcourus. Pour cela, il faudra supprimer la restriction qu'un nœud ne doit pas être visité deux fois. Cette modification dans l'algorithme est faite aux dépens de son efficacité et à la condition que des mécanismes complexes

[13]En anglais : *crawling*.

d'identification de cycles soient inclus. C'est peut-être pour ces raisons que certaines des approches de découverte d'unités logiques d'information proposées jusqu'à présent construisent d'abord le graphe complet sous-jacent à l'ensemble de pages à analyser avant de commencer la procédure de détermination des unités logiques d'information, i.e. les documents.

Avant de présenter les différentes méthodes proposées dans la littérature, nous résumons brièvement la problématique abordée. La prémisse de base est qu'une page HTML ne contient pas une information auto-explicative : par le seul contenu d'une page un acteur ne peut pas appréhender toute l'information qu'elle véhicule, il existe d'autres pages qui complètent ou contextualisent le contenu de cette page. La problématique est de proposer une méthode pour identifier ces pages complémentaires.

Dans cette étude il est question d'analyser l'utilité de la connaissance des pages complémentaires dans le cadre d'un moteur de recherche. Dans ce cadre, l'identification des documents sert à indexer plus précisément les pages. L'index d'une page p_x ne doit pas être uniquement basé sur le contenu de la page, $Cont(p_x)$, mais aussi sur le contenu des pages qui lui sont complémentaires, $Cont(Compl(p_x))$:

$$Index(p_x) = f(Cont(p_x), Cont(Compl(p_x))) \tag{2.1}$$

Dans ce qui suit nous présentons une liste d'aspects qui permettent d'organiser les principales méthodes de découverte de pages complémentaires proposées jusqu'à l'heure actuelle. Après nous présentons brièvement les méthodes classées selon l'un de ces aspects et finalement nous présentons un tableau comparatif des différentes méthodes selon les caractéristiques que nous avons dégagées.

2.2.2 Aspects des méthodes

Afin de mieux présenter et comparer l'ensemble des méthodes d'identification d'unités logiques dans le Web proposées jusqu'à ce jour nous avons identifié sept aspects inhérents à toutes les méthodes. L'ensemble des aspects crée une base commune de comparaison puisque chaque aspect peut être traité différemment selon la méthode. Les aspects sont les suivants : la granularité de la réponse du moteur de recherche utilisant la méthode, le nombre d'unités logiques d'information (ULI) par page, le moment de découverte du graphe, le moment de découverte des ULIs, le type d'information utilisée pour la découverte des ULIs, la prise en compte de l'hétérogénéité des liens sur le Web et la localité de l'information utilisée pour la découverte des ULIs.

Parmi les critères qui ont été choisis, nous pouvons identifier des critères généraux qui ne sont pas spécialement attachés à l'application aux moteurs de recherche : par exemple, le nombre d'unités logiques d'information (ULI) par page et le type d'information utilisée pour la découverte des ULIs. En revanche d'autres critères n'ont un sens que dans le contexte de l'application spécifique aux moteurs de recherche, e.g. la granularité de la réponse du moteur de recherche et le moment de la découverte des ULIs (avant ou après la requête). Il est évident que s'il était question d'analyser les méthodes dans le contexte d'un autre type d'application, nous pourrions nous servir d'autres aspects de comparaison plus spécifiques à l'application

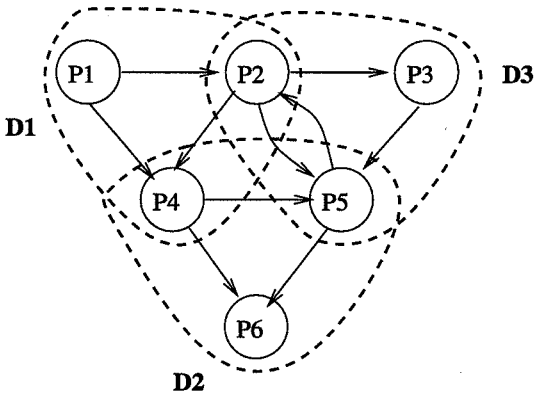

FIG. 2.5: Regroupement de pages dans des documents.

en question. Par exemple, si l'application visée était la visualisation d'un système hypertexte nous pourrions définir un aspect lié au type de structure de l'hypertexte généré par les ULIs découvertes[14]. La structure peut être plus ou moins artificielle selon la méthode utilisée. Par exemple nous pouvons imaginer une méthode dans laquelle le nombre maximum de ULIs générées et/ou le nombre maximum de pages dans chaque ULI seraient prédéfinis.

Dans ce qui suit nous détaillons chacun des aspects énumérés plus haut. Pour chacun des aspects nous identifions les différentes manières de le traiter tout en précisant les avantages et les inconvénients des différentes manières.

2.2.2.1 La granularité de la réponse du moteur de recherche

Ce critère concerne le type de réponse renvoyée par le moteur de recherche en réponse à une requête. En général nous pouvons identifier deux types de réponse : une liste de pages ou une liste d'unités logiques d'information où chaque unité logique est composée d'une ou plusieurs pages.

Évidemment, la réponse du moteur dépend de l'unité d'information qu'il a indexée. Si le moteur a indexé des pages, il ne peut renvoyer qu'une liste de pages en réponse à une requête. Inversement, si l'indexation se fait au niveau des unités logiques d'information, la réponse sera probablement composée d'une liste de telles unités. Cependant, nous pouvons concevoir qu'un moteur qui indexe des unités logiques d'information puisse renvoyer aussi des pages. Une fois qu'une ULI pertinente à la requête a été trouvée, le moteur peut décider de retourner une page composant l'ULI et non pas l'ULI entière. Des raisons pour un tel choix existent et seront évoquées dans la suite.

Considérons le graphe illustré par la figure 2.5 et analysons l'indexation de la page p_4. Cette page fait partie de deux documents, le document D_1 et le document D_2. Imaginons

[14] À partir d'un hypertexte dont les nœuds représentent des pages nous pouvons construire un deuxième hypertexte où les nœuds représenteraient les ULIs découvertes du premier hypertexte. Les liens entre les ULIs seraient déduits des liens entre les pages composant les ULIs.

maintenant les deux cas de figures possibles selon que le système indexe les pages ou les documents. Dans le premier cas de figure, l'index de la page p_4 doit être fonction de son contenu et de celui des pages qui lui sont complémentaires :

$$Index(p_4) = f(Cont(p_4), Cont(p_1), Cont(p_2), Cont(p_5), Cont(p_6))$$

Par contre, dans le deuxième cas de figure les documents sont indexés et non pas les pages. Ainsi nous retrouvons les index suivants :

$$Index(D_1) = f(Cont(p_1), Cont(p_2), Cont(p_4))$$

$$Index(D_2) = f(Cont(p_4), Cont(p_5), Cont(p_6))$$

$$Index(D_3) = f(Cont(p_2), Cont(p_3), Cont(p_5))$$

Admettons maintenant que, pour une requête donnée, le document composé par les pages p_4, p_5 et p_6 soit pertinent. Si nous sommes dans le premier cas de figure le système doit renvoyer comme réponse la liste des pages p_4, p_5 et p_6 dans un ordre donné. En revanche, si nous sommes dans le deuxième cas de figure, le système doit rendre comme réponse le document D_2 ou l'une des pages le composant. Quels sont les avantages et les inconvénients de ces approches ?

Un utilisateur comprend mieux une réponse du type page qu'une réponse du type document (ensemble de pages). La raison est que les utilisateurs sont habitués à des réponses composées de pages, style adopté par tous les moteurs traditionnels. Il leur est plus difficile d'interpréter une réponse constitué d'un graphe de pages, voire d'un ensemble de pages, c.-à-d. une réponse multi-entité et non pas mono-entité. Réciproquement, l'utilisateur pourra avoir des difficultés pour comprendre la raison pour laquelle une page a été renvoyée comme réponse alors qu'il se peut que les mots clés de sa requête ne soient pas tous inclus dans le contenu la page en question. Une solution possible à ces problèmes de compréhension serait de développer une interface de visualisation de réponses qui permette à l'utilisateur de comprendre pourquoi les réponses retournées ont été jugées pertinentes à la requête soumise.

L'avantage de l'indexation des documents est la cohérence entre la granularité de l'indexation et la structure de l'information qui est stockée dans l'hypertexte.

Par contre, nous pouvons signaler un inconvénient assez important du choix de l'indexation de pages. Prenons encore comme exemple l'index de la page p_4. Son index est fonction du contenu de plusieurs pages qui ne sont pas forcément toutes complémentaires les unes des autres. D'après le document D_1, la page p_4 est complémentaire à la page p_2, ainsi, le contenu de ces deux pages doit être vu comme un ensemble. D'autre part, d'après le document D_2, la page p_4 est complémentaire à la page p_6 et par conséquent, le contenu de ces deux pages doit être vu comme un autre ensemble. Cependant, lorsque l'index de la page p_4 est considéré comme étant une fonction du contenu de plusieurs pages, deux contenus qui n'ont pas forcément une relation de complémentarité entre eux, en l'occurrence, celui de la p_2 et celui de p_6, sont mis ensemble ou mélangés (cf. figure 2.5). Un effet néfaste de ce mélange est,

qu'en réponse à une requête, p_4 soit jugée pertinente à cause de l'association du contenu de p_2 et p_6. Pour éviter ce genre de problème, tout en gardant l'option d'indexer des pages, il faudrait que l'index possède une certaine structure de sorte que le contenu de p_2 ne soit pas mélangé avec celui de p_6. Par exemple, nous pourrions imaginer deux index distincts pour p_4 : le premier $Index(p_4)'$ concernerait le document D_1, et le deuxième $Index(p_4)''$ concernerait le document D_2. L'inconvénient de cela est l'augmentation de la complexité de la mise en œuvre du module d'indexation. Notons que dans l'exemple donné, l'évaluation de la pertinence de p_4 serait faite deux fois. À la première fois p_4 serait considérée comme composante du document D_1 ($Index(p_4)'$ serait utilisé) et à la deuxième fois p_4 serait considéré comme composante du document D_2 ($Index(p_4)''$ serait utilisé).

$$Index(p_4)' = g(Cont(p_4), Cont(p_1), Cont(p_2))$$

$$Index(p_4)'' = g(Cont(p_4), Cont(p_5), Cont(p_6))$$

Finalement, par rapport à l'espace nécessaire pour stocker l'index, nous pouvons imaginer que l'indexation par pages soit plus gourmande que l'indexation par documents. La raison de cela est que le contenu d'une page (ou une partie) est inclus dans l'index de toutes les pages qui lui sont complémentaires. Dans le dernier exemple, en dehors de son propre index, le contenu de la page p_4 serait utilisé dans l'index des pages p_1, p_2, p_5 et p_6. Évidemment, on peut faire en sorte que les index soient factorisés et que les termes d'indexation dérivés du contenu de p_4 soient stockés dans un seul emplacement. Ensuite, il faudra que l'index des pages complémentaires à p_4, en l'occurrence p_1, p_2, p_5 et p_6, fasse référence à l'index de p_4. Dans ce cas, lors de l'évaluation de la pertinence d'une page, la méthode de récupération de cet index sera plus complexe et l'exécution plus lente puisque il faudra aller chercher des parties d'index ailleurs.

Nous avons dit plus haut qu'au premier abord, l'indexation par pages est plus gourmande que l'indexation par documents. Ce n'est pourtant pas toujours vrai. En effet, l'indexation par documents peut être autant, voire plus, gourmande que l'indexation par pages, cela puisqu'une page peut faire partie de plusieurs documents à la fois. De ce fait, l'index correspondant au contenu de la page sera dupliqué dans l'index de tous les documents dont la page fait partie. Dans le dernier exemple, l'index de la page p_4 (ou une partie) sera inclus dans le document D_1 et D_2 (cf. figure 2.5). En effet, plus il y a des pages faisant partie de plusieurs documents à la fois, plus gourmande en termes d'espace devient l'indexation. Le nombre de pages possédant cette caractéristique dépend de deux facteurs : (i) de la méthode employée pour identifier les documents et (ii) du style de rédaction utilisé pour créer les pages. En ce qui concerne le premier facteur, la méthode employée pour découvrir les ULIs peut faire en sorte qu'aucune page n'appartienne à plusieurs documents (voir la prochaine section). Dans cette situation, il n'y a pas besoin d'espace supplémentaire pour l'indexation. En ce qui concerne le deuxième facteur, dans un graphe où les documents seraient très peu fractionnés la situation décrite au-dessus arriveraient moins souvent que dans un graphe où les documents seraient fractionnés davantage. En revanche, il est vrai que si les documents sont très fractionnés, la factorisation de l'index devient possible permettant ainsi d'envisager une réduction de l'espace nécessaire pour stocker les index. Cependant, comme nous l'avons cité plus haut, la factorisation est faite aux dépens du temps de calcul nécessaire pour qu'un

moteur réponde aux requêtes. Nous retrouvons ici le fameux compromis entre le temps de calcul et l'espace rencontré en algorithmique.

2.2.2.2 Le nombre d'unités logiques d'information (ULI) par page

Ce critère concerne le nombre d'unités logiques auxquelles une page peut appartenir. Dans l'exemple de la figure 2.5, nous pouvons apercevoir qu'à l'instar de page p_4, les pages p_2 et p_5 appartiennent à deux documents différents chacune. Ainsi, la méthode d'identification des documents utilisée dans l'exemple permet qu'une page appartienne à plus d'un document. Cependant, on peut imaginer d'autres méthodes qui ne permettent pas qu'une page appartienne à plus d'un document. Le premier type de méthode génère donc des documents non-disjoints et le deuxième génère des documents disjoints.

Quels sont les avantages et inconvénients des deux types de méthodes ? Intuitivement, nous pouvons imaginer qu'une page puisse appartenir à deux documents différents. Une méthode qui contraint une page à appartenir à une seule ULI impose une structure artificielle sur l'information analysée. S'il n'est pas question de cette contrainte, on peut espérer que la structure naturelle de l'information soit révélée. Encore faut-il que la méthode appliquée pour identifier les documents soit efficiente.

Les méthodes qui génèrent des documents non-disjoints doivent avoir aussi une demande de ressources en termes d'espace supérieure à celle des méthodes disjointes et cela pour la raison évoquée à la fin de la dernière section concernant la granularité de la réponse des moteurs de recherche. En plus, ces méthodes présentent normalement une complexité de calcul qui est supérieure à celle des méthodes disjointes. La raison de cela est que les méthodes non-disjointes imposent moins de contraintes afin de révéler une structure plus naturelle des données. On peut faire une analogie avec la distinction entre les méthodes disjointes et non-disjointes de clusterisation. Dans [vR79, page 44], Rijsbergen signale : « *Dans le processus de clusterisation, il y a une certaine quantité d'information qui est rejetée ou ignorée de sorte que tous les membres d'un même cluster soient indifférenciables les uns des autres. C'est dans l'objectif de tenter de minimiser la quantité d'information écartée, de trouver des clusters étant en quelque sorte le plus proche possible des données originelles, que les clusters non-disjoints sont acceptés. Malheureusement, cela est fait aux dépens de l'efficacité et de l'efficience de l'implémentation.* »

2.2.2.3 Le moment de découverte du graphe

Toutes les méthodes de découverte d'ULIs proposées jusqu'à présent utilisent la structure de liens entre les pages pour mesurer la complémentarité existant éventuellement entre elles. La relation floue qu'un lien établit entre deux pages du Web est effectivement supposée être une éventuelle indication de complémentarité. Afin d'explorer les liens entre les pages analysées pour la méthode d'identification d'ULIs, il faut que le graphe reliant les pages soit disponible. Le graphe peut être découvert avant ou après la soumission de la requête.

L'inconvénient majeur de construire le graphe en aval de la soumission de la requête est la répercussion sur le temps de réponse du moteur de recherche. Si le réseau est très chargé ou, pire, si le réseau est indisponible au moment où la requête est posée, le temps nécessaire pour que le moteur retourne sa réponse sera inadmissible pour l'utilisateur. En plus du temps

requis pour télécharger les pages, il faut évidemment considérer le temps d'analyser syntaxiquement les pages pour trouver leurs liens, le temps pour découvrir les ULIs, pour indexer leur contenu et finalement pouvoir évaluer la pertinence des ULIs par rapport à la requête.

Cependant une découverte en aval possède aussi ses avantages. Premièrement, l'information utilisée pour découvrir les ULIs sera fraîche, mise à jour. Nous savons que l'information existante dans le Web est très volatile est que, sur certains sites, les pages, et par conséquent, le graphe, peuvent changer plusieurs fois dans une même journée[15].

Le deuxième avantage d'une découverte en aval est le fait que le système n'a pas besoin de disposer d'une quantité de ressources conséquente pour son fonctionnement. Les index des pages ou des ULIs sont construits après la formulation d'une requête. Ils sont utilisés pour évaluer la pertinence de l'information qu'ils représentent par rapport à la requête. Leurs identifiants sont retournés à l'utilisateur en réponse et, ensuite, les index sont aussitôt détruits. A priori, les index ne sont pas persistants. Évidemment, selon la taille du graphe à parcourir lors de chaque formulation de requête, l'approche en aval est plus ou moins faisable[16].

Un dernier point à préciser à propos de la méthode de découverte en aval est celui concernant le(s) nœud(s) de départ du parcours. Autrement dit, à partir de quelle(s) page(s) la découverte du graphe commence ? Le robot responsable du parcours utilise une liste de pages comme nœuds de départ. Cette liste de pages n'est bien entendu pas générée aléatoirement. Les URLs incluses dans cette liste sont censées représenter des pages situées dans des régions du graphe contenant des pages présentant une forte probabilité d'être sinon pertinentes, au moins liées au besoin d'information exprimé dans la requête.

2.2.2.4 Le moment de découverte des ULIs

À l'instar du dernier aspect, celui-ci se positionne par rapport au moment où la requête est posée. Il s'agit du moment de la découverte des unités logiques d'information. Un premier

[15] Cette volatilité des pages dans le Web est d'ailleurs l'un des grands problèmes rencontrés par les moteurs de recherche exhaustifs (dont le but est d'indexer le Web visible entier) telles qu'*altavista* et *google*. Il s'agit d'un problème lié à la fraîcheur des index. Le temps écoulé entre deux visites successives d'un robot à une URL donnée peut aller jusqu'à des dizaines de jours. Si le contenu sous une URL est indexé à la date t_{index1}, si la page est modifiée à la date t_{modif} et la prochaine visite du robot ne se fait qu'à la date t_{index2}, entre la date t_{modif} et la date t_{index2}, l'index de la page située sous l'URL donnée est faux, c.-à-d. l'index ne correspond pas au contenu de la page. Ainsi, entre les dates t_{modif} et t_{index2}, tous les jugements faits sur le contenu lié à l'URL donnée sont inexacts. L'effet de cela ne peut être que néfaste pour l'efficience du moteur. Si la page en question est jugée pertinente pour des requêtes posées entre les dates t_{modif} et t_{index2}, renvoyer son URL en réponse correspond à donner une mauvaise réponse à la requête posée. Cela augmente le taux de bruit de la réponse. De l'autre côté, si la page sous l'URL donnée est jugée non-pertinente pour toutes les requêtes entre ces mêmes deux dates, il est vrai que le bruit ne se fait pas entendre. Cependant, au cas où par manque de chance le nouveau contenu associé à l'URL donnée est pertinent à certaines des requêtes posées entre les deux dates, la non-fraîcheur de l'index aura contribué à l'augmentation du taux de silence du moteur.

[16] Une manière d'éviter que le temps de parcours du graphe devienne trop long est de déterminer un temps maximal de parcours qui, une fois atteint déclenche l'arrêt du du parcours. Alternativement, une profondeur maximale de parcours par rapport à la page où le parcours débute peut aussi contrôler indirectement le temps qui est dédié à la découverte du graphe. Cependant, ces deux approches — temps maximal ou profondeur maximale — ont le même inconvénient qui est celui de limiter arbitrairement le parcours. Rien ne permet de faire a priori une corrélation entre le temps ou la profondeur maximale et la qualité de la réponse qui sera retournée à l'utilisateur.

point à signaler est que ce critère est bien-entendu très lié au critère précédent puisque la découverte des ULIs est basée sur le graphe des pages. Ainsi, pour que la découverte des ULIs soit faite il faut que le graphe ait déjà été construit. Une méthode qui découvre les ULI avant la requête et le graphe après la requête n'a donc pas de sens. Par contre, une méthode qui découvre les ULI après la requête peut découvrir le graphe aussi bien en amont qu'en aval de la requête.

Les avantages et les inconvénients de la découverte des ULIs en amont ou en aval de la requête sont les mêmes que ceux évoqués dans le critère précédent. D'un côté la découverte en amont permet au moteur de recherche de garantir un temps de réponse court aux requêtes posées. En plus, puisque la découverte des ULIs n'est faite que rarement[17] le système peut se permettre de faire appel à une méthode de découverte d'ULIs plus robuste et consommatrice d'un temps de calcul important. Un dernier avantage de la découverte en amont des ULIs est qu'elles peuvent être utilisées dans des modules annexes au moteur de recherche de sorte à aider l'utilisateur dans sa tâche de récupération d'information. Par exemple, les ULIs peuvent être utilisées pour construire une carte du graphe sous-jacent à l'information traitée par le moteur mais à une granularité différente de celle des pages. Ce niveau de granularité supplémentaire permet de réduire la dimensionnalité de l'espace d'information en passant d'une *granularité page* à une *granularité ULI*. L'espace de dimension réduite peut donner à l'utilisateur une meilleure compréhension générale de l'information sur laquelle sa recherche se fait et ainsi, peut lui permettre de mieux traduire son besoin d'information sous forme de requête. Les inconvénients sont les ressources nécessaires pour stocker les index de l'information manipulée et la non-fraîcheur de cette information.

2.2.2.5 Le type d'information utilisée pour la découverte des ULIs

Cet aspect concerne le type de source d'information exploitée par la méthode de découverte d'ULIs. D'une manière générale, nous pouvons identifier trois sources principales d'information dans le Web : le contenu des pages, la structure entre les pages et le(s) fichier(s) qui garde(nt) des informations concernant les accès aux serveurs HTTP utilisés pour héberger les pages[18].

On trouve une classification analogue dans les travaux concernées par le domaine du *Web Mining* [BL99] [MBNL99]. Dans ce domaine, les applications sont classées selon le type de données qu'elles utilisent. Nous retrouvons les termes *Web Content Mining*, le *Web Structure Mining*, le *Web Usage Mining*[19]. Nous étudions maintenant ces trois sources d'information par rapport à notre besoin de découvrir la complémentarité entre les pages.

Le contenu

Nous entendons par contenu d'une page le tuple suivant :

$$p = (url, mime\_type, taille, date\_derniere\_modification, texte)$$

[17] La découverte des ULIs n'est faite qu'à des intervalles de temps importants de sorte à rafraîchir l'information manipulée.

[18] À partir de ce moment, nous allons appeler ce fichier *fichier de log*.

[19] Bien entendu, le *Web Mining* est pour le processus de *Knowledge Discovery in the Web* ce que le *Data Mining* est pour le processus de *Knowledge Discovery in Databases*.

Évidement il s'agit d'une représentation simplifiée d'une page. On pourrait y rajouter d'autres attributs tels que la date d'expiration de la page et le jeu de caractères utilisés. Nous avons fait le choix d'utiliser ces attributs puisque ce sont ceux-là que les méthodes citées dans la prochaine section utilisent.

Dans le domaine de la linguistique il existe des travaux concernant l'identification du contexte d'un texte. Cependant, ces travaux ne s'adaptent pas bien à notre cas pour deux raisons principales : (i) les documents ne sont pas clairement identifiés dans un hypertexte ; (ii) le style de rédaction dans un hypertexte est un style à part entière et différent de celui observé dans des documents non fractionnés.

Dans les méthodes qui utilisent le contenu comme indicateur de complémentarité, le contenu est souvent utilisé pour supporter ou renforcer un deuxième indicateur basé sur une autre propriété des données analysées. Par exemple, si la structure qui relie deux pages données donne des fortes indications qu'elles sont complémentaires, l'analyse du contenu des pages peut être utilisé pour renforcer ou affaiblir ces indications.

La structure

Le deuxième type d'information utilisable dans l'analyse de complémentarité entre les pages est la structure qui les relie, c.-à-d. les liens. Nous avons déjà cité que la grande difficulté dans l'exploitation des liens dans le Web est lié au fait qu'ils ne sont pas explicitement typés et à la diversité de types. Nous pouvons considérer un lien comme étant un tuple :

$$l = (url\_source, url\_destination, texte\_ancre, type\_lien)$$

L'attribut *type_lien* peut avoir comme valeur : *lien intra-site* ou *lien inter-site*. Sa valeur est déduite des valeurs *url_source* et *url_destination*.

Remarquons ainsi que non seulement le fait que deux pages soit liées entre elles peut être un indicateur de complémentarité mais aussi la structure de répertoire codée dans l'URL le texte de l'ancre et le type du lien peuvent l'être. L'inconvénient de l'utilisation des URLs est que les méthodes d'organisation des fichiers représentant les pages dans un système de fichiers peuvent varier beaucoup selon les auteurs des pages. Dans un extrême, nous pouvons imaginer toutes les pages d'un site stockées dans un même répertoire, dans l'autre extrême nous pouvons imaginer pour chaque page un répertoire différent, l'ensemble des répertoires étant organisé hiérarchiquement.

En ce qui concerne le texte de l'ancre des liens, il est difficile d'imaginer d'autres manières de l'utiliser que celle basée sur une liste de termes susceptibles d'indiquer que la page pointée est complémentaire de la page où le lien se trouve. Par exemple, on peut admettre que les liens incluant dans le texte de l'ancre les termes *continuer* ou *prochain* indiquent une relation de complémentarité. Encore une fois, le style de rédaction de l'auteur influe beaucoup sur l'efficacité d'une telle méthode. Une étude intéressante à propos de l'analyse du style de rédaction des ancres des liens afin de déterminer les liens d'un type spécifique a été publiée dans [Ami00].

En plus de la structure reliant les pages, la structure inter-pages, nous pouvons aussi envisager l'utilisation des structures intra-pages de deux pages pour chercher des indications de complémentarité entre les pages. Cependant, la tâche ne semble pas être facile. D'une part, il faudrait trouver une manière d'identifier deux structures qui se complètent en quelque sorte. Même si l'on définit une mesure d'association/distance entre deux structures internes, son efficience serait sans doute peu élevée. Cela est due à la difficulté de la découverte de la structure logique d'une page. Il est vrai que certains éléments du HTML, e.g. `<H1>..<H6>`, `<P>`, ont été initialement prévus pour donner une structure logique[20] aux pages, malheureusement dans la pratique, ce type d'élément est utilisé pour la mise en forme des pages plutôt que pour expliciter leur structure logique. De ce fait, l'identification de la structure logique d'une page peut s'avérer très difficile.

L'usage

Le troisième type de donnée potentiellement exploitable pour découvrir les ULIs est la trace de l'usage/consultation des serveurs HTTP utilisés pour héberger les pages à travers l'analyse de ce que nous avons appelés plus haut *fichiers de log*. Par ailleurs, certains sites demandent à leurs visiteurs de s'enregistrer en fournissant des données concernant leur profil lors de leur première visite au site. Nous pouvons concevoir l'utilisation des profils et des données extraites des fichiers de *log* dans une méthode de découverte d'ULIs.

En ce qui concerne le fichier de *log*, selon le serveur HTTP utilisé, le fichier peut comporter des informations différentes. Toutefois, d'une manière globale, les informations suivantes sont toujours disponibles dans le fichier de *log* quelque soit le type de serveur : l'URL de la page demandée par le client, le numéro d'IP de la machine du client, la date et l'heure où la demande a été faite. Certains clients sont configurés de sorte que l'URL de la page éventuellement affichée[21] dans la fenêtre client qui a déclenché la requête soit aussi envoyée au serveur HTTP. Une autre information susceptible d'être retrouvée dans une entrée du fichier de *log* est le nom de l'application utilisée par le client pour accéder au serveur, cependant les clients sont libres de décliner leur identité ou non [FIG+97]. De son côté, le serveur peut être configuré de sorte que cette information soit enregistrée dans l'entrée du fichier de *log* correspondant à la demande. Il s'agit d'une information intéressante puisqu'elle facilite la trace d'un même utilisateur à l'intérieur d'un site. Cependant, cette donnée n'est pas toujours disponible.

En ce qui concerne la découverte des ULIs, le type d'information que l'on cherche à identifier dans le fichier de *log* est le(s) chemin(s) parcouru(s) par un utilisateur pendant une session. Certains de ces chemins — ceux empruntés le plus fréquemment par des différents utilisateurs — peuvent représenter des suites de pages composant des unités logiques d'information.

Les difficultés de cette analyse sont considérables, de plus, une série d'hypothèses sont faites par les méthodes qui font ce genre d'analyse. L'hypothèse principale consiste à considérer que la navigation d'un utilisateur reflète un type de lecture où les documents sont lus entièrement. Ainsi, si un document est composé d'un chemin regroupant cinq pages, l'utilisa-

[20] En opposition à la structure physique des pages, communément appelée mise en forme des pages.

[21] Cette donnée ne peut exister que si l'application client utilisée pour demander une page est un navigateur ou un robot.

teur n'est pas censé arrêter sa lecture à la deuxième page et dériver sur une page concernant un autre document. Or, dans la section 1.3.3, nous avons évoqué un type de désorientation cognitive particulière aux systèmes hypertextes appelée « *les digressions imbriquées* » par Le Crosnier [LC95]. Cette désorientation est due aux cheminements marginaux empruntés par les utilisateurs pendant leur navigation à cause d'un changement de centre d'intérêt. D'autre part, le fait que les pages d'un site ne soient pas bien conçues font que l'utilisateur ne dispose pas d'assez d'indicateurs pour identifier les suites logiques d'une page. Cela peut mener l'utilisateur à un changement de centre d'intérêt involontaire.

Ceci dit, le changement de centre d'intérêt, et par conséquent, de document pendant la navigation peut être aussi volontaire. Un utilisateur qui serait plutôt intéressé à avoir une vision générale du site ne rentrerait pas trop dans les détails de chaque document mais en ferait une *lecture en largeur* plutôt qu'en profondeur.

Étant donné ce problème de changement de centre d'intérêt lors du parcours des utilisateurs au sein d'un site, une idée pour identifier les vrais chemins consisterait à identifier pour chaque page ses successeurs les plus probables lors d'un parcours d'utilisateur. À partir d'un fichier de *log* on pourrait obtenir ce type d'observation. D'ailleurs, il est probable qu'il existe une corrélation entre le nombre de visites enregistrées dans le fichier et l'efficacité de l'analyse. Une fois les séquences de pages susceptibles de former des vrais chemins identifiées, on pourrait appliquer des heuristiques basées sur la similarité entre le contenu des deux pages successives dans un chemin, sur la structure de répertoire codée dans l'URL des pages, etc., afin de sélectionner les chemins qui seraient considérés comme étant des unités logiques d'information.

En supposant vraie l'hypothèse selon laquelle l'utilisateur ne change pas souvent de document pendant sa navigation au sein d'un site, il reste un ensemble de difficultés à traiter lors de l'analyse d'un fichier de *log*. Dans la suite nous énumérons un certain nombre d'entre elles.

Premièrement, il faut être capable d'identifier les requêtes HTTP envoyées par des robots de moteurs de recherche pour ne pas en tenir compte. En effet, dans la plupart du temps, les robots exécutent un parcours qui ne correspond pas à la lecture d'un document entier à chaque fois[22]. Les parcours sont généralement exécutés en profondeur ou en largeur sans vraiment tenir à parcourir des suites des pages ayant entre elles une relation identifiable. Les noms des robots de quelques moteurs de recherche sont publiquement connus : par exemple celui d'*Altavista* s'appelle *scooter* et celui de *Google* se nomme *Googlebot* [And]. Certains de ces robots, lors de l'envoi du message HTTP pour demander une page, s'identifient en remplissant le champ "User-Agent" [FIG+97] avec leur nom. Une autre possibilité consiste à vérifier le nom symbolique de la machine ayant envoyé le message HTTP. Très souvent, dans le nom symbolique de la machine on peut identifier le nom du robot. Malheureusement, ni l'une ni l'autre de ces règles ne sont impératives et donc ce ne sont que des heuristiques qui essayent d'identifier lorsqu'une demande HTTP est faite par un robot et non pas par un utilisateur humain.

[22] L'hypothèse citée au-dessus ne concerne que le comportement de navigation d'acteurs humains et non pas d'acteurs logiciels.

La deuxième grande difficulté est d'identifier si deux requêtes HTTP sont émises par un même utilisateur ou non. En effet, l'IP d'une machine n'identifie pas toujours un unique utilisateur. De plus en plus souvent un pare-feu sépare un ensemble de machines reliées en intranet de l'extérieur. Dans ce cas, la requête d'une machine dans l'intranet passe par le pare-feu avant d'arriver au serveur. Le pare-feu peut être configuré de sorte que tous les paquets TCP qui sortent de l'intranet aient les bits correspondant à l'IP de la machine originaire du paquet modifiés. Dans ce cas, lorsque le message part du pare-feu vers le serveur HTTP, ce n'est plus l'adresse IP de la machine client mais l'adresse du pare-feu qui figure dans le paquet TCP. Cela peut donc cacher plusieurs utilisateurs différents derrière une même adresse IP. Dans ce cas aussi, il faut des heuristiques au niveau de l'application d'analyse de *logs* pour distinguer les chemins d'utilisateurs différents se trouvant derrière un pare-feu.

Il semble que les méthodes d'analyse de fichiers de *log* soient indissociables des heuristiques. À notre connaissance, aucune méthode de découverte d'ULIs basée sur l'analyse des fichiers de *log* n'a été proposée jusqu'à ce jour. Dans [PPR96] une analyse d'un fichier des *log* est faite mais dans le but de découvrir des points de repère — des pages — dans le site associé au fichier, ceci pour aider l'utilisateur dans la visualisation et la navigation du site. Certains des problèmes liés à l'analyse des fichiers de *log* évoqués plus haut sont discutés dans cet article.

Finalement, une dernière difficulté — et non des moindres — de l'analyse des fichiers de *logs* est due à la disponibilité de ces fichiers. Il se peut que la personne ou l'organisation propriétaire du serveur HTTP associé au fichier de *log* à analyser ne souhaite pas rendre public ce fichier pour une raison ou pour une autre.

2.2.2.6 La prise en compte de l'hétérogénéité des liens sur le Web

Nous avons déjà dit plus haut que l'une des caractéristiques du Web était la diversité des types de liens, un lien exprimant une relation implicite d'un type particulier entre deux pages. Par exemple, nous pouvons imaginer des liens reflétant une relation de structure logique entre deux pages (*page chapitre* liée à plusieurs *pages section*) ; une relation de structure séquentielle de lecture des pages (page A pointant sur la page B indiquant que la lecture de la page A doit être suivie de la lecture de la page B) ; un lien de publicité ; un lien de citation (dans le sens bibliographique), etc.

Il est en fait difficile de définir une taxonomie des types de liens. D'une part la frontière entre les différents types risque d'être extrêmement floue. Par exemple, jusqu'où un lien de citation n'est pas un lien de publicité et vice-versa ? D'autre part, dû au caractère multi-auteur du Web qui se traduit par l'utilisation de ce média de publication par des acteurs de plus en plus différents, ayant des besoins différents par rapport au type d'information à publier, on constate que la relation exprimée par les liens est presque *ad-hoc* selon l'information manipulée et son type ne rentre pas exactement dans un des types basiques cités plus haut. En revanche, cela n'est pas une raison pour que la communauté scientifique travaillant sur le Web n'essaie pas de faire une première typologie par rapport au type de relation exprimée par les liens entre les pages du Web.

La diversité de types de liens en tant que telle ne doit évidemment pas être considérée comme étant une caractéristique néfaste du Web mais plutôt comme une richesse de ce mé-

dia. Pour ce qui concerne les moteurs de recherche, le vrai inconvénient réside dans le fait que les liens ne sont pas explicitement typés. Nous avons déjà évoqué plus haut la difficulté d'établir une typologie et, ensuite, imposer aux auteurs des pages d'utiliser cette typologie dans la création des liens. Nous pensons que si l'on considère le Web en général, le typage *a priori* des liens, i.e. pendant leur création, est complètement utopique. Pourtant, les typologies des liens pourraient être définies dans le cadre d'un type d'application spécifique, par exemple l'application moteur de recherche. Une idée consiste à établir une typologie qui serait utile dans le cadre de l'application et ensuite développer des heuristiques qui permettraient de classer les liens entre les pages automatiquement selon la typologie définie. Nous verrons une approche utilisant cette idée dans la section 2.2.3.2.

D'après les types basiques cités plus haut, en termes de découverte des ULIs nous pensons que les liens qui permettraient de faciliter l'identification des ULIs représentées par un ensemble de pages seraient les liens de structure logique et les liens de navigation. Cependant, nous savons que si d'un côté tous les liens de structure logique sont potentiellement utiles pour la découverte d'ULIs dans un site Web, d'un autre côté uniquement certains liens de navigation le sont. Alors qu'un lien de navigation reliant deux pages qui doivent être visitées l'une à la suite de l'autre peut s'avérer très utile, un lien de navigation menant chaque page d'un document à la première page de ce document, p.ex. sa table de matières, peut nuire à l'efficacité d'une méthode de découverte d'ULIs basée sur ce type de lien. De cette façon, même si les liens étaient typés selon ces deux classes, les difficultés pour trouver les ULIs seraient aussi importantes.

L'aspect traité dans cette section permettra donc de distinguer les méthodes de découverte d'ULIs qui considèrent tous les liens comme étant similaires de celles qui tiennent compte du fait qu'ils ont des types différents. Ces dernières méthodes essaient d'identifier les liens qui peuvent leur être utiles dans le processus de découverte d'ULIs. Pour faire la distinction et le tri des liens, les méthodes peuvent aussi bien se baser sur le contenu des pages reliées par les liens que sur les données relatives au lien lui-même (cf. la section précédente). Par exemple, une méthode de découverte d'ULIs peut être modélisée de sorte que pour une page donnée, disons p_x, seuls les liens intra-site soient exploités dans le processus de recherche des pages complémentaires de p_x. Pour identifier ce type de lien la méthode se basera sur la valeur du paramètre *type_de_lien* contenu dans le tuple représentant le lien. Un autre exemple de distinction de type de liens peut être une méthode qui se base sur la structure de répertoire codée dans l'URL des deux pages reliées par le lien analysé. Si l'un des répertoires est un sous-répertoire de l'autre le lien peut être considéré comme étant d'un certain type alors que si les fichiers correspondant aux deux pages se trouvent dans le même répertoire les liens peuvent être considéres comme étant d'un autre type. Une méthode qui tiendrait compte de l'hétérogénéité des liens pourrait utiliser uniquement l'un de ces deux types de liens comme indicateur de complémentarité entre deux pages.

2.2.2.7 La localité de l'information composant une ULI

Ce critère spécifie le type d'information composant une ULI en termes de localité. Admettons que nous voulions trouver l'information complémentaire au contenu de la page p_i située dans le site S_x. L'information complémentaire peut se trouver soit dans d'autres pages

situées dans le même site que p_i, i.e. dans S_x, soit dans des pages situées dans d'autres sites. Nous allons appeler le premier type d'information *information locale* et le deuxième type *information non locale*. Ce critère permettra donc de distinguer les méthodes qui n'utilisent que de l'information locale de celles qui utilisent de l'information locale et de l'information non-locale pour découvrir les ULIs.

Le seul avantage qu'une méthode utilisant de l'information non locale peut avoir sur une méthode qui n'utilise que de l'information locale est éventuellement la quantité supérieure d'information disponible à explorer. Intuitivement, plus on a de l'information concernant une page donnée, plus riche et complet peut être le contexte de cette page. Le contexte que l'on trouve dans d'autres pages situées dans le même site que la page analysée peut être considéré comme étant un contexte local. Ce contexte local peut être considéré comme étant une vue ou une interprétation de l'information par son auteur. Certes il s'agit d'une interprétation légitime mais, à l'extérieur du site, nous pouvons trouver des pages apportant d'autres contextes à l'information, contextes qui peuvent être considérés comme d'autres vues et/ou d'autres interprétations de l'information. Ces contextes auxiliaires peuvent être bien différents de celui imaginé/proposé par l'auteur de la page et, par conséquent, utiles dans l'indexation.

En revanche, l'utilisation d'information non locale possède quelques inconvénients. Premièrement, l'information non locale est beaucoup plus chère en termes de temps consacré à sa récupération que l'information locale. Lorsqu'il est question d'information locale, l'application qui mène à la découverte des ULIs des pages d'un site est souvent exécutée soit dans la même machine que le serveur HTTP hébergeant le site soit dans une machine proche physiquement, souvent faisant partie du même sous-réseau. Dans cette situation, le téléchargement des pages est rapide puisque le nombre de pages à charger est réduit et la vitesse de chargement est élevée.

Inversement, lorsqu'il est question d'utiliser l'information non locale, la méthode de découverte d'ULIs peut s'avérer beaucoup plus coûteuse. Si d'une part le téléchargement des pages demande beaucoup d'effort puisque les serveurs abritant les pages demandées peuvent être très distants de la machine où l'application responsable du téléchargement s'exécute, d'autre part la définition des pages à télécharger peut être aussi délicate. Considérons par exemple une méthode qui utilise les liens entrants et sortants d'une page — donc ses voisins immédiats — comme information de base pour la découverte des pages complémentaires à une page donnée, disons p_i. Pour identifier les pages pointées par p_i il suffit de faire l'analyse syntaxique de son contenu en identifiant les liens qui partent d'elle. Par contre, pour identifier les pages qui pointent sur p_i ce n'est pas aussi simple. Il faut disposer du graphe représentant toutes les pages susceptibles de complémenter p_i. Or, comme nous sommes dans la situation où l'information non locale est utilisée, les pages susceptibles de complémenter le contenu de p_i sont toutes les pages du Web. Ainsi, la méthode doit disposer du graphe complet du Web. À notre connaissance, il n'existe pas d'application qui prétende avoir disposé à un instant donné d'une copie du graphe du Web entier. La difficulté de se procurer une telle copie est dûe (i) au nombre inconnu de pages existantes ce qui rend difficile la prévision des ressources nécessaires ; (ii) au caractère distribué du Web et (iii) à sa volatilité. Finalement, les conditions instables du trafic sur Internet peuvent rendre impossible le parcours entier du Web.

Supposons qu'au lieu du graphe complet, une sous-partie du graphe soit suffisante pour l'analyse. Dans ce cas, nous pouvons envisager d'utiliser les services d'autres moteurs pour trouver l'information souhaitée, en l'occurrence, les pages qui pointent sur p_i. En effet, certains moteurs gardent la structure du graphe qu'ils ont parcouru et fournissent à l'utilisateur des moyens d'interroger cette structure. Par exemple, pour connaître l'ensemble de pages qui pointent sur une page donnée, *Google* et *Altavista* proposent dans leur langage de requête l'opérateur *link:*. Ainsi, si l'utilisateur veut récupérer toutes les pages parcourues par le robot d'*Altavista* ayant un lien sur une page dont l'URL est url_i, il doit saisir la requête *link* : url_i.

On peut imaginer une modification de la méthode dans la mesure où il est question de récupérer non seulement les voisins immédiats de p_i mais aussi les voisins se trouvant à une distance de deux liens de p_i. L'explosion des requêtes du type *link* : *url* à envoyer à *Altavista* peut être remarquable selon la page dont on veut récupérer le voisinage. Dans ce cas, le temps consacré à envoyer la requête et recevoir la réponse ajouté à celui du téléchargement des pages fait que le temps total nécessaire pour l'exécution d'une telle méthode deviendrait prohibitif. Et cela même si l'analyse est faite en amont de la requête. Avec l'accroissement exponentiel du nombre de pages dans le Web et le trafic de plus en plus lourd, de telles méthodes deviennent difficiles à envisager.

Le deuxième inconvénient d'une méthode utilisant l'information non locale pour trouver les ULIs est l'incertitude de trouver l'information complémentaire d'une page donnée dans des pages situées dans d'autres sites que celui où la page analysée se trouve. En revanche, il est quasiment certain que l'information complémentaire peut être trouvée dans le même site que celui de la page analysée. En effet, ce n'est que dans des situations exceptionnelles qu'un auteur rédigeant un document hypertexte place deux pages concernant le même document dans deux sites différents[23].

2.2.3 Principaux travaux classés selon le moment de définition de l'unité logique

Après avoir présenté les différents aspects d'une méthode de découverte d'ULIs, nous choisissons l'un d'entre eux pour passer en revue les différentes approches proposées jusqu'à ce jour : le moment de la découverte des ULIs. Comme nous avons vu plus haut cette découverte peut être faite en amont ou en aval de la requête. Avant de passer à la suite nous tenons à faire deux remarques.

Premièrement, les premiers travaux concernant cette problématique datent de 1998. Ils ne sont pas très nombreux, dépassant à peine une dizaine. Nous avons effectué une recherche exhaustive et dans la suite nous présentons les travaux fondamentaux. La deuxième remarque à faire est que certains travaux cités dans la suite abordent des problématiques qui ne sont pas toujours celle de la découverte d'ULIs bien que très proches. D'autre part, même si certaines méthodes ont comme but la découverte d'ULIs, l'usage de ces ULIs n'est pas toujours dans le contexte d'un moteur de recherche. Lorsque ce sera le cas, nous le signalerons.

[23] Il est encore question de ce que l'on appelle *site*. Nous aborderons cette question dans la section 3.1.

Dans ce qui suit nous présentons les approches principales proposées jusqu'à ce jour concernant la découverte d'ULIs dans les pages du Web. Nous commençons par les approches où les ULIs sont découvertes en amont de la requête et, dans la section suivante, les méthodes exécutées en aval de la requête sont présentées. Les avantages et les inconvénients de la découverte des ULIs en amont et en aval de la requête ont été évoquées dans la section 2.2.2.4.

Bien entendu, cette étude a pour but de dégager les caractéristiques communes et particulières de chaque méthode afin de mieux situer et justifier notre propre méthode.

Avant la requête

Dans la suite nous passerons en revue les méthodes qui découvrent les ULIs avant la requête.

2.2.3.1 La méthode des coupures d'un graphe

Dans [TMKT98], Tajima propose une méthode de découverte d'ULIs qui se base aussi bien sur le contenu des pages que sur les liens entre elles. Dans cette méthode, le graphe de départ sous-jacent aux pages du Web appartenant à un même site est amputé des arcs entre les nœuds associés à des pages abordant deux thèmes différents. Sur ce nouveau graphe, les *cuts* — un *cut* est une composante connexe du graphe partiel résultant de l'amputation d'arcs du graphe original — sont considérées comme étant les ULIs. L'hypothèse sous-jacente à la méthode est donc que toute paire de pages complémentaires doivent être reliées par une chaîne. La méthode utilise la similarité entre le contenu de deux pages pour déterminer si elles concernent le même thème ou non. Cette fois l'hypothèse sous-jacente est que le niveau de similarité entre le contenu de deux pages est plus élevé si elles discutent un même thème, hypothèse reprise du travail de Hearst et al. [HP93].

Les méthodes proposées dans l'article sont génériques et peuvent être appliquées à n'importe quel type de donnée hypertextuelle dans la mesure où elles se basent sur le contenu textuel des nœuds et sur les liens, les deux types de données que l'on retrouve dans n'importe quel système hypertexte. Les données hypertextuelles considérées dans l'article sont des pages HTML, des articles de forums de discussion et des méls[24]. Des ensembles de messages issus de forums de discussion ou d'*e-mails* peuvent être considérés comme des systèmes hypertextes dont les nœuds représenteraient les messages et les liens entre les nœuds seraient déduits des valeurs des champs *In-Reply-To* et *References* trouvés dans l'en-tête des messages [Cro82][25].

Deux méthodes sont proposées dans [TMKT98]. Toutes les deux utilisent le modèle vectoriel pour représenter le contenu textuel des nœuds. Les poids des termes sont calculés d'après une variante de la mesure $tf \times idf$[26]. Un nœud est représenté par le vecteur :

$$V(n_i) = (w_{i1}, ..., w_{im}) \tag{2.2}$$

[24] Messages d'une boîte aux lettres électroniques, en anglais *e-mails*.

[25] Cela dit, ces deux champs sont optionnels et chaque *mailer* et/ou lecteur de *newsgroups* peut choisir de traiter ou non ces champs.

[26] *Term frequency* × *inverse document frequency* : type traditionnel de formule de pondération utilisé dans les systèmes de recherche d'information. Le poids d'un terme dans un document est directement proportionnel à sa fréquence d'apparition dans le document et inversement proportionnel au nombre de documents où il apparaît.

où w_{ik} est le poids du terme d'indice k dans le document n_i et m est le nombre de termes retrouvés dans le contenu de l'ensemble des nœuds de l'hypertexte considéré. La mesure de similarité entre le contenu de deux nœuds, disons n_i et n_j, est le cosinus entre les deux vecteurs représentant les contenus :

$$Sim(n_i, n_j) = \frac{V(n_i) \cdot V(n_j)}{|V(n_i)||V(n_j)|} \tag{2.3}$$

Les deux méthodes suggérées utilisent cette mesure. En ce qui concerne la première méthode, elle est fondée sur l'algorithme de clusterisation hiérarchique lien-moyen[27]. L'algorithme est modifié afin de tenir compte du fait que les *cuts* doivent représenter des composantes connexes du graphe sous-jacent aux pages analysées. Ainsi, pendant l'exécution de l'algorithme, deux entités (un nœud ou un ensemble de nœuds) sont regroupées à la condition qu'il existe au moins un lien entre un nœud dans l'une des deux entités relié par un lien à un nœud de l'autre entité. L'algorithme associé à la première méthode est résumé ci-dessous :

- Entrées : un graphe orienté $G = (N,E)$ où N est l'ensemble de nœuds et E l'ensemble d'arcs et n correspond au nombre de *cuts* dans lequel le graphe doit être partitionné.

- Soit S un ensemble de $|N|$ *singleton cuts*, chaque *cut* contenant un nœud de N
- Tant que le nombre d'éléments de S est supérieur à n :
  - Calculer la similarité $Sim(Cut_i, Cut_j)$ pour tous les $Cut_i, Cut_j \in S$ tel que $\exists d_i \in Cut_i, \exists d_j \in Cut_j$ et $(d_i, d_j) \in E$
  - Sélectionner la paire Cut_i, Cut_j dont $Sim(Cut_i, Cut_j)$ est maximale, remplacer Cut_i et Cut_j dans S par Cut_k où $Cut_k = Cut_i \cup Cut_j$

Pour utiliser l'algorithme ci-dessus il faut disposer d'une formule qui permette de calculer la similarité entre deux *cuts*, l'équation 2.3 permettant de calculer uniquement la similarité entre deux *cuts* de type singleton. La formule proposée dans l'article pour calculer cette similarité est :

$$Sim(\{n_1, \cdots, n_p\}, \{n'_1, \cdots, n'_q\}) = \frac{V(\{n_1, \cdots, n_p\}) \cdot V(\{n'_1, \cdots, n'_q\})}{|V(\{n_1, \cdots, n_p\})||V(\{n'_1, \cdots, n'_q\})|} \tag{2.4}$$

où $V(\{n_1, \cdots, n_p\}) = \sum_{1 \leq i \leq p} V(n_i)$ représente une sorte de centroïde du *cut* composé des nœuds $n_1 \cdots n_p$. Notons que le critère d'arrêt de l'algorithme est fonction du nombre de *cuts* déjà générés à un instant donné. Une alternative serait de fixer une similarité minimale entre deux *cuts* en deçà de laquelle plus aucun regroupement ne pourrait avoir lieu. Il est évident que l'un des inconvénients de cet algorithme est ce critère d'arrêt arbitraire. En effet, ce critère impose une structure artificielle à l'ensemble de pages traitées et ne révèle pas une structure naturelle des données. Autrement dit, il est difficile d'expliquer pourquoi un nombre maximal de clusters donné serait plus adéquat qu'un autre. Les auteurs n'ont pas cité des résultats qui permettraient une évaluation de l'efficience de l'algorithme. En revanche, après avoir testé l'algorithme sur un ensemble de nœuds de forums de discussion, deux problèmes

[27] Plus connu sous son nom en anglais : *group-average link*.

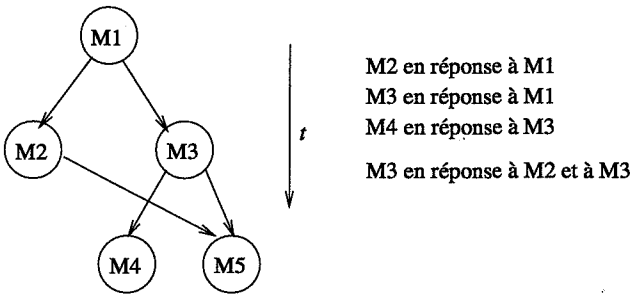

FIG. 2.6: Exemple de création de messages dans le temps dans un forum de discussions.

généraux ont été détectés.

Le premier problème consiste dans le fait qu'un nœud représentant un message dans un forum de discussion doit pouvoir appartenir à deux *cuts* différents. L'algorithme décrit plus haut génère des *cuts* disjoints. En effet, un nœud intermédiaire peut inspirer l'apparition d'autres nœuds, ou des *threads*, qui aborderont des sous-thèmes de l'information traitée dans le nœud intermédiaire — nœud M_1 inspirant l'apparition des nœuds M_2 et M_3 (cf. figure 2.6). D'autre part, nous pouvons concevoir qu'un nœud puisse apparaître dans la suite de deux autres nœuds (apparition du nœud M_5 inspirée par les nœuds M_2 et M_3).

La deuxième observation que les auteurs ont faite est que si les nœuds intermédiaires d'un fil de messages d'un forum de discussions contiennent une information souvent générale, multi-thématique, les nœuds feuille, du moins, ceux situés vers le bas de la hiérarchie[28] sous-jacente au fil de messages analysés, contiennent des informations plus spécifiques. Or, le comportement observé de l'algorithme montre que les nœuds intermédiaires se regroupent dans un même *cut* tôt dans l'exécution. La conséquence de cela est la réduction du nombre de regroupements entre les nœuds intermédiaires et les nœuds feuilles auxquels ils sont liés thématiquement.

Ayant observé les deux problèmes du premier algorithme, les auteurs en ont proposé un deuxième. Le nouvel algorithme essaye de résoudre les problèmes de la manière suivante : (i) il permet qu'un nœud fasse partie de plusieurs *cuts* à la fois et (ii) les *cuts* commencent à être formés à partir des feuilles ce qui doit permettre qu'un nœud contenant de l'information que les auteurs qualifient de spécifique soit regroupé avec les nœuds intermédiaires contenant de l'information concernée par le même thème.

Ce deuxième algorithme part du principe qu'en entrée on dispose d'un arbre. L'hypothèse de base est que plus on descend dans l'arbre, plus spécifique est l'information contenue dans les nœuds. Ainsi, le nœud contenant l'information la plus générique est le nœud racine et les feuilles contiennent les informations les plus spécifiques. Étant donné que ni les données

[28] La structure sous-jacente à un fil de messages d'un forum de discussions représente une hiérarchie puisqu'il y a toujours un nœud racine (le nœud contenant le message qui a déclenché le *thread*). Par contre, selon l'interprétation des champs *In-Reply-to* et *References* éventuellement présents dans l'en-tête des messages, la structure peut être un arbre ou non.

des forums de discussions, ni les données de la messagerie, ni les données du Web ne forment un arbre au départ, il faut pré-traiter les données pour transformer la structure qui leur est sous-jacente dans celle d'un arbre avant l'application de l'algorithme.

En ce qui concerne les forums de discussions et la messagerie, les nœuds possédant plus d'un père sont dupliqués. En ce qui concerne les données du Web, le pré-traitement est composé de deux étapes. La première consiste à identifier le(s) nœud(s) racines. Le nœud racine est censé être celui ayant une URL du genre `http://aaa.bbb.ccc/index.(s)htm(l)`[29]. Si un tel nœud n'existe pas, tous les nœuds se trouvant dans le répertoire racine, i.e. sous `http://aaa.bbb.ccc/`, sont considérés comme étant des nœuds de type racine. La deuxième étape consiste à construire la hiérarchie une fois que le(s) nœud(s) racine(s) a(ont) été identifié(s). Pour cela, un parcours en largeur est effectué. Les liens menant à des pages qui ont déjà été visitées sont ignorés. Finalement, pour ce qui concerne les données du Web, les auteurs ont considéré qu'une unité logique d'information est composée uniquement de pages d'un même site. Ainsi, le parcours en largeur que nous venons de citer ne prend en compte que les liens internes, i.e. les liens menant à des pages situées dans le même site que la page racine. Dans cette étude, deux pages appartiennent à un même site si leurs URLs ont un même préfixe[30].

Les problèmes relevés par les auteurs qui ont motivé la réflexion sur un deuxième algorithme ont été détectés sur des tests effectués sur des données issues de forums de discussions. Des tests sur d'autres types de données hypertextuelles n'ont pas été faits, du moins les auteurs ne les ont pas cités. Ainsi, nous ne disposons pas d'indications si les mêmes problèmes se reproduisent avec les données du Web ou avec celles de la messagerie. Si le premier problème semble être bien susceptible de se reproduire avec des données différentes de celles issues d'un forum de discussion, pour ce qui du deuxième problème, celui lié à la généralité (resp. spécificité) des nœuds intermédiaires (resp. feuilles), il est difficile de l'imaginer se reproduire dans d'autres contextes, surtout dans le Web où la notion de nœuds intermédiaires et de nœuds feuilles n'existe pas et que le graphe sous-jacent aux pages n'est pas hiérarchisé contrairement à la configuration des données relevant de la messagerie ou d'un forum de discussion. De plus, alors que dans les deux derniers types de données un nœud racine est facilement identifié, dans le Web, ce type de nœud n'existe pas systématiquement et, lorsqu'il existe, il ne peut pas toujours être identifié par une procédure automatique.

L'hypothèse considérée par l'algorithme est que dans un arbre généré à partir d'un nœud racine tel que les auteurs l'ont défini et d'un parcours en largeur où les liens menant à des pages déjà visitées sont ignorés, les nœuds situés vers le haut contiendraient des informations plus génériques que celles contenues dans les nœuds situés vers les feuilles. Aucun résultat de l'application du deuxième algorithme n'a été donné dans l'article. Ainsi, nous ne pouvons pas savoir si l'hypothèse considérée se vérifie ni combien l'algorithme est efficient dans la découverte des ULIs dans un arbre généré à partir de l'application de l'hypothèse.

L'application proposée pour les ULIs découvertes par l'algorithme est celle d'un moteur de recherche. Au lieu d'indexer des pages, le moteur de recherche indexe des ULIs. Ainsi, les

[29] Les URLs `http://aaa.bbb.ccc/welcome.(s)htm(l)` et `http://aaa.bbb.ccc/home.(s)htm(l)` sont d'autres exemples de ce même genre.

[30] Dans la section 3.1 on aborde la problématique de la définition d'un site.

requêtes soumises par les utilisateurs sont comparées avec des ULIs et non pas avec des pages. Cependant, des précisions n'ont pas été données sur la manière dont les ULIs sont indexées (certaines pages sont-elles plus représentatives du contenu d'une ULI que d'autres ? Quelles parties des pages sont prises en comptes pour l'index ?) ni sur la manière dont la pertinence d'une ULI à une requête donnée est calculée. D'autres applications ont aussi été évoquées : la clusterisation des réponses d'un moteur de recherche ou encore celle des messages d'un forum de discussion.

2.2.3.2 La méthode des chemins d'entrée des pages

Dans ce qui suit nous présentons une deuxième étude de Tajima [MT99] ayant comme problématique le fractionnement des documents dans un système hypertexte. Contrairement à l'étude que nous venons de détailler, dans cette deuxième étude il est question de trouver des unités logiques d'information seulement dans un hypertexte constitué de pages HTML. Les graphes sous-jacents aux messages de forums de discussions et *e-mail* ne sont pas considérés.

Comme nous l'avons évoqué dans la section 2.2.1, pendant l'écriture d'une page donnée, disons p_x, son auteur suppose qu'au moment où elle sera lue par un lecteur, celui-ci aura déjà parcouru un chemin de pages qui lui aura fourni le contexte ou le prérequis pour comprendre l'information véhiculée par p_x. Dans ce travail, ce chemin de pages que l'utilisateur a dû parcourir avant d'arriver à p_x et qui a été conçu — consciemment ou non — par l'auteur de p_x est appelé *le chemin d'entrée* de p_x. L'idée consiste donc à retrouver le chemin d'entrée de chaque page d'un site. Les pages composant le chemin d'entrée de p_x sont considérées comme étant l'unité logique d'information à laquelle appartient p_x.

Pour découvrir le chemin d'entrée de chaque page, une méthode basée sur l'URL des pages et les liens entre elles est proposée. Contrairement à la méthode des coupures d'un graphe, le contenu des pages n'est pas utilisé. L'algorithme sur laquelle la méthode suggérée se base est composé de deux étapes principales. La première étape consiste à trouver *le lien d'entrée* pour chaque page du site analysé. Le lien d'entrée d'une page donnée est le lien par lequel un lecteur est censé arriver à la page. La deuxième étape de l'algorithme consiste à faire la *concaténation* de chaque page avec celle d'où son lien d'entrée émane (figure 2.7). Bien entendu la difficulté de la méthode réside dans la première étape, i.e. trouver le lien d'entrée d'une page donnée.

L'hypothèse liée à la première étape considère qu'une page possède, au plus, un lien d'entrée. Par conséquent, il est supposé qu'un lecteur arrive toujours à une page donnée par un chemin unique. Cela implique qu'une page HTML est censée appartenir à une seule unité logique d'information. Cependant, tel que nous l'avons vu plus haut, il est raisonnable de considérer qu'une page puisse appartenir à plus d'une unité logique d'information. Les auteurs reconnaissent que la l'hypothèse qu'une page soit associée à un seul chemin d'entrée représente une approximation grossière de la réalité et que l'idéal serait bien entendu d'avoir une méthode qui découvre tous les éventuels chemins d'entrée d'une page. Le lien d'entrée d'une page est choisi parmi les n liens candidats — les liens pointant sur la page — selon une probabilité déterminée d'après un nombre d'hypothèses que nous citons dans la suite.

L'hypothèse liée à la deuxième étape établit que si le chemin ch_i est le chemin d'entrée d'une

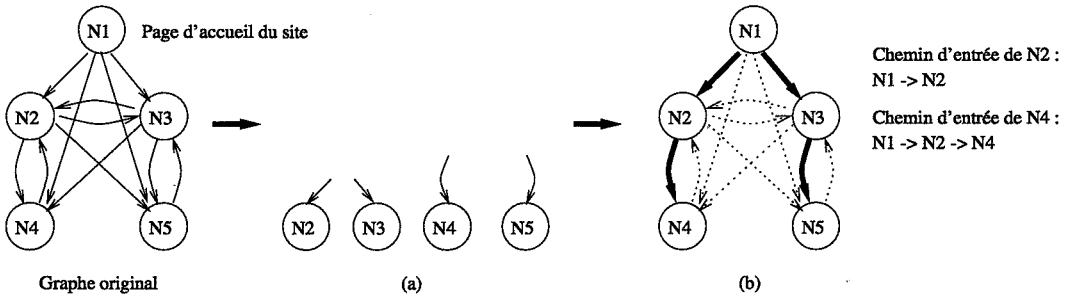

FIG. 2.7: Deux étapes de la découverte des chemins d'entrées des pages : (a) découverte du lien d'entrée de chaque page, (b) concaténation des liens d'entrée.

page p_i et l, lien émanant de p_i, est le lien d'entrée de la page p_j, le chemin d'entrée de la page p_j est le chemin résultant de la concaténation de ch_i et de l. Il est important de souligner que cette hypothèse ne se vérifie pas systématiquement dans la réalité. Par exemple, considérons que p_i et p_j aient été créées par deux auteurs différents. Dans ce cas, même si l'auteur de p_j considère le lien l comme le lien d'entrée de la page p_j, il y a de fortes chances que ce même auteur ne sache pas que l'auteur de la page p_i l'a créée en supposant ch_i comme son chemin d'entrée. En effet, les auteurs justifient l'utilisation de cette hypothèse simplificatrice par une question de complexité de calcul de la méthode. L'hypothèse réduit le coût de la calcul de la méthode dans la mesure où le problème de la découverte des chemins d'entrée de toutes les pages d'un site se réduit au problème de la découverte du lien d'entrée de chaque page du site.

En ce qui concerne les deux étapes de l'algorithme, nous allons présenter ici juste les idées principales concernant la première étape. L'implémentation de la deuxième étape est triviale.

Le premier pas de la première étape consiste à identifier tous les liens vers la page dont on veut découvrir le lien d'entrée et à les classer selon la typologie suivante : lien intra-répertoire, lien vers-le-bas, lien vers-le-haut, lien transverse et lien inter-site[31]. Cette typologie est établie selon les URLs des deux pages connectées par le lien analysé :

- lien intra-répertoire : lien entre deux pages stockées[32] dans un même répertoire ;
- lien vers-le-bas : lien émanant d'une page stockée dans un répertoire R_x vers une page stockée dans un sous-répertoire de R_x ;
- lien vers-le-haut : lien émanant d'une page stockée dans un répertoire R_x vers une page stockée dans un répertoire ancêtre de R_x[33] ;
- lien transverse : lien entre deux pages stockées dans deux répertoires, sans relation de parenté, situés dans un même site Web[34] ;
- lien inter-site : lien entre deux pages hébergées par deux sites Web différents.

[31] En anglais respectivement : lien *intradirectory*, *downward*, *upward*, *sibling* et *intersite*.
[32] À partir de ce moment, par une question de simplicité, lorsque nous écrirons *une page stockée dans un répertoire donné*, on doit comprendre *le fichier représentant une page et stocké dans un répertoire donné*.
[33] Le répertoire /dirA/ est un répertoire ancêtre du répertoire /dirA/dirB/dirC.
[34] Les répertoires /dirA et /dirB situés tous les deux dans un même site Web constituent deux répertoires sans relation de parenté.

Ensuite, une typologie logique des liens est proposée selon leur rôle dans le site. Les trois types définis sont :

- lien d'entrée : lien qui sert de lien d'entrée à une page donnée ;
- lien de retour : lien émanant d'une page vers une autre page située dans le chemin d'entrée de la première ;
- lien de saut : lien n'étant ni un lien d'entrée ni un lien de retour.

Alors que la première typologie est objective la deuxième est complètement subjective. Une première série d'hypothèses telles que : « *Un lien du type vers-le-bas ne représente jamais un lien du type retour* » est proposée pour établir une correspondance entre les types de liens de la première typologie et ceux de la deuxième typologie. Ces hypothèses permettent d'identifier les candidats à devenir lien d'entrée d'une page donnée.

Après avoir identifié les candidats à devenir lien d'entrée d'une page à l'aide de la première série d'hypothèses, l'algorithme en applique une deuxième série qui permet d'identifier parmi tous les liens candidats celui étant le plus probable d'être le lien d'entrée de la page analysée. La première hypothèse de ce deuxième jeu d'hypothèses établit que s'il y a plusieurs candidats à devenir le lien d'entrée d'une page donnée, les liens du type vers-le-bas ont une probabilité plus grande d'être lien d'entrée que les liens des autres types. Les liens de type intra-répertoire sont classés en deuxième rang, ensuite viennent les liens de type inter-site et finalement les liens de type transverse. Par exemple, même s'il existe un seul lien de type vers-le-bas parmi les candidats mais plusieurs candidats de type intra-répertoire, c'est le lien du type vers-le-bas qui est sélectionné comme lien d'entrée.

Des hypothèses additionnelles sont prévues pour résoudre les situations où (i) il existe plusieurs liens du type vers-le-bas candidats à devenir lien d'entrée d'une page donnée; (ii) il n'existe pas de liens du type vers-le-bas candidats mais il existe plusieurs candidats du type intra-répertoire; (iii) il n'existe pas de liens du type intra-répertoire mais il en existe plusieurs de types inter-site et transverse. Ces dernières hypothèses permettent de choisir un lien parmi plusieurs liens ex æquo.

Finalement, une fois que parmi tous les liens candidats un seul a été choisi, il faut encore décider s'il est vraiment retenu comme le lien d'entrée de la page ou non. Cette décision se fait à l'aide de deux dernières hypothèses : la première concerne le cas où le lien candidat est soit du type vers-le-bas soit du type intra-répertoire. La deuxième hypothèse concerne le cas où le lien candidat est soit du type transverse soit du type inter-site. Si le lien candidat à devenir le lien d'entrée d'une page donnée n'est finalement pas retenu comme tel, la page est considérée comme étant une page d'entrée du site, i.e. une page sans lien d'entrée. Une page d'entrée de site est une page dont la lecture n'est pas supposée être faite en aval de la lecture d'autres pages du même site auquel la page appartient. Par exemple, la page d'accueil d'un site[35] constitue très souvent une page d'entrée de site. Cependant, nous pouvons imaginer d'autres pages d'entrée. Par exemple, considérons un ensemble de pages concernant une conférence hébergé par un serveur HTTP associé à un centre de recherche et que le chercheur qui organise la conférence travaille dans ce centre. Ce chercheur a inséré un lien dans sa page

[35] Page ayant une URL du type *http://aaa.bbb.ccc/index.htm(l)*.

personnelle vers une page qui sert d'index à l'ensemble de pages concernant la conférence. Cette page index peut être considérée comme étant une page d'entrée de site. Il est vrai que l'ensemble de pages concernant la conférence peut être considéré comme un site à part entière selon la définition que l'on adopte pour un *site* (dans la section 3.1 nous revenons sur cette question).

Dans [MT99], la méthode de découverte des chemins est proposée dans le but d'améliorer l'indexation de pages et, par conséquent, d'augmenter l'efficacité d'un moteur de recherche. Nous pouvons imaginer l'ensemble de pages composant le chemin d'entrée d'une page donnée et de la page elle-même comme étant une unité logique d'information. Cependant l'unité d'information traitée par le moteur de recherche proposée dans l'article n'est pas celle d'une ULI mais celle d'une page. L'utilisation de l'information concernant l'ULI à laquelle une page donnée, disons p_i appartient est faite de la manière suivante : pour construire l'index de p_i le module d'indexation se base non seulement sur le contenu de p_i mais aussi sur le contenu des pages se trouvant dans le chemin d'entrée de p_i.

Il reste à déterminer si l'indexation de p_x doit considérer le contenu complet des pages faisant partie du chemin d'entrée de p_x ou seulement certaines parties. Dans la deuxième option, il faut encore sélectionner quelles parties devraient être considérées. Alors que l'on peut imaginer des situations où la récupération partielle du contenu des pages serait la décision adéquate à prendre, nous pouvons imaginer aussi d'autres situations où récupérer le contenu complet des pages serait plus adéquat. L'idéal serait donc de faire une analyse cas par cas et prendre la décision la plus adéquate selon le cas. Évidemment, une telle analyse demanderait à identifier dans chaque page du chemin d'entrée la ou les parties qui complètent/contextualisent la page qui suit dans le chemin d'entrée. À notre avis, une telle méthode ne pourrait pas être envisageable sans l'intégration d'une analyse linguistique poussée. À défaut d'une telle méthode, une décision arbitraire entre prendre en compte le contenu complet ou seulement une partie du contenu des pages du chemin d'entrée doit être faite. Dans [MT99], la décision a été de prendre en compte seulement certaines parties du contenu des pages, à savoir le contenu des balises `<title>`, `<a>` et `<h1> ... <h6>`. En ce qui concerne la balise `<a>`, seule le texte de l'ancre du lien qui constitue le lien d'entrée en question est pris en compte et non pas les ancres de tous les liens de la page. Il en est de même pour les balises `<h1> ... <h6>`, seul le texte de l'en-tête de la section qui contient le lien d'entrée est pris en compte.

La méthode de découverte des chemins d'entrée des pages a été testée sur un site Web contenant 840 pages. La méthode a correctement découvert les liens d'entrée de 78% des pages analysées. Cependant le moteur de recherche utilisant la méthode d'indexation proposée dans le dernier paragraphe n'a pas été implémenté. Ainsi, aucun résultat démontrant l'efficacité de la méthode d'indexation dans l'amélioration de la qualité des réponses d'un moteur de recherche n'a été donné.

La méthode proposée est basée sur un nombre conséquent d'hypothèses, soit 16 au total. Toutefois, à notre avis, les hypothèses ne sont pas toujours justifiées d'une manière convaincante. En plus, elles dépendent très fortement d'une hypothèse majeure qui porte sur le style d'organisation des fichiers représentant les pages dans un système de fichiers. Il est supposé que l'organisation utilisée par l'auteur des pages est hiérarchique. Si la méthode est appliquée

à un site où les fichiers représentant les pages ne s'organisent pas hiérarchiquement, nous citons les auteurs : «*la méthode proposée risque de détecter un chemin d'entrée incorrect pour presque toutes les pages du site.* » Or, il est difficile de justifier la méthode sans évaluer le pourcentage de sites dans le Web adoptant un style d'organisation de pages qui la met en échec.

Par ailleurs, l'association d'un chemin d'entrée à chaque page d'un site est certes une idée très intéressante, cependant, il reste à prouver que l'auteur des pages place la complémentarité d'une page donnée dans le chemin d'entrée de cette page. Il semble évident que pour une page donnée, disons p_x, il existe d'autres pages dans le même site que p_x qui complètent son contenu. Cependant, affirmer que ces pages se trouvent dans le chemin d'entrée de p_x semble être moins évident. Nous pensons également que les pages complémentaires de p_x se trouvent dans son voisinage (en termes de la structure du graphe), cependant nous croyons que ce voisinage s'identifie par une organisation un peu plus floue que celle d'un chemin de nœuds dans le graphe sous-jacent au site. Nous expliquerons cette supposition dans le chapitre 3 où nous développons notre méthode.

2.2.3.3 La méthode de l'identification des sous-domaines logiques dans un site Web

L'étude décrite dans [LKV00] propose une méthode pour la découverte des domaines logiques à l'intérieur d'un même domaine physique du Web. Un domaine physique est considéré comme étant composé de toutes les pages dont les URLs partagent le nom de la machine d'accueil et le numéro de port. Ainsi, les pages représentées par les URLs `http://www.emse.fr:80/~aguiar/publications.htm` et `http://www.emse.fr/~aguiar/biography.htm` font partie d'un même domaine physique. Notons cependant qu'une même machine d'accueil peut avoir plusieurs noms, ce que l'on appelle les *aliases* de la machine. Ainsi, si `www1.emse.fr` constitue un `alias` de la machine `www.emse.fr`, les deux machines sont associées à une même adresse IP. Ce cas n'est pas évoqué dans [LKV00]. Par conséquent, deux pages associées aux URLs `http://www.emse.fr/1.html` et `http://www1.emse.fr/2.html` sont considérées appartenir à deux domaines physiques différents dans l'algorithme que nous citons dans cette section.

En contrepartie, un domaine logique est défini comme étant un ensemble de pages situées dans un même domaine physique du Web formant un ensemble cohérent d'information, ce que nous avons appelé plus haut *unité logique d'information.* Dans un site d'un département de recherche nous pouvons par exemple associer un domaine logique à l'ensemble des pages personnelles d'un chercheur, à l'ensemble des pages concernant une conférence organisée par le département, etc.

Un domaine logique peut donc être consideré comme étant une ULI telle que nous l'avons définie dans la section 2.2.1. Nous y avons précisé qu'une ULI, de par sa définition, pourrait avoir des granularités différentes. La granularité des unités d'information visée dans [LKV00] est moins fine que celle visée dans tous les autres articles cités dans cette section. Un domaine logique peut être vu comme une unité logique d'information pouvant être divisée en unités logiques d'information ayant une granularité équivalente à celle considérée dans les autres méthodes évoquées dans cette section. Malgré la différence de granularité nous citons

ce travail à cause de la proximité entre la problématique qui y est abordée et la nôtre, les méthodes et conclusions issues de cette étude pouvant être utiles dans le développement de méthodes de découverte d'ULIs de granularité plus fine.

L'objectif de la découverte des ULIs dans [LKV00] ne se trouve pas dans l'amélioration de la qualité des index utilisés par un moteur de recherche. La motivation de cette étude est de proposer une meilleure organisation de la liste d'URLs renvoyée par un moteur de recherche en réponse aux requêtes qui lui sont soumises. L'idée est de proposer à l'utilisateur une liste de clusters d'URLs à la place d'une liste d'URLs. Dans un cluster sont regroupées des URLs associées à des pages appartenant à un même domaine logique. À l'aide des clusters, l'utilisateur doit pouvoir analyser plus efficacement les réponses en rejetant les clusters qu'il juge être hors sujet[36].

Une façon efficace de clusteriser les résultats consiste à regrouper les URLs par nom de machine d'accueil (cf. le moteur de recherche *google*). Cependant, cette technique ne fonctionne pas très bien avec des sites très larges et hétérogènes thématiquement comme *geocities* (`www.geocities.com`), *multimania* (`http://membres.lycos.fr`), *w3c* (`www.w3c.org`), etc. Un site de ce type contient des pages qui peuvent traiter de thèmes très différents[37] et qui ainsi ne doivent pas se retrouver dans un même cluster lorsque renvoyées en réponse par un moteur de recherche. D'où l'idée de découvrir les différents domaines logiques d'un site. Une fois que les domaines logiques ont été découverts pour un site donné, si dans la liste de réponses d'un moteur de recherche ayant indexé ce site il figure deux pages faisant partie d'un même domaine logique, ces pages sont rendues ensemble dans un cluster.

L'algorithme proposé pour la découverte des domaines logiques est composé de deux étapes principales. La première consiste à découvrir les pages d'entrée des domaines logiques du site. La deuxième consiste à établir les frontières des domaines logiques en partant des pages d'entrée découvertes dans l'étape précédente.

En ce qui concerne la première étape, la page d'entrée d'un domaine logique est définie comme étant la page racine[38] du domaine, i.e. la première page qu'un utilisateur est censé visiter lorsqu'il parcourt le domaine logique en question. La découverte des pages d'entrée des domaines logiques d'un site est basée sur l'application d'un ensemble de règles à chacune des pages composant le domaine physique analysé. Chaque page est associée à un score qui représente les chances qu'elle soit reconnue comme page racine d'un domaine logique. Chaque règle augmente ou diminue le score d'une page. Les règles se basent sur le contenu textuel des pages, plus précisément sur le contenu de la balise `<title>`, la structure des liens entre les pages, le texte de l'ancre des liens et l'URL des pages. Un sous-ensemble des règles proposées est illustré dans la figure 2.8.

[36] Les clusters dont les pages ne sont pas pertinentes au besoin d'information exprimé par la requête.

[37] Intuitivement, cela doit aussi se constater dans de sites ayant un nombre de pages réduit mais à une moindre échelle.

[38] La structure du graphe sous-jacent à un domaine logique ne constitue pas forcément un arbre, ainsi le qualificatif *racine* utilisé dans ce contexte n'a pas la même sémantique que celle associée au nœud racine d'un arbre.

```
Règle #1    url                                    : " /~[^/]*/?$ "          : +60
Règle #2    url  : " ^[^~]*/(people|users?|classe(s)|projects?|seminars?)/$ "  : +30
Règle #5    title                                  : " \bhome\b "            : +10
            title                                  : " \bwelcome\b "         : +5
Règle #6    texte de l'ancre de lien rentrant      : " ^home$ "              : +5
Règle #7    texte de l'ancre de lien sortant       : " ^home$ "              : -10
Règle #9    nombre de liens externes rentrants     : > 0                     : +20%
Règle #10   nombre de liens sortants               : > 20                    : +5
```

FIG. 2.8: Sous-ensemble des règles utilisées pour classer les pages candidates à page d'entrée des domaines logiques d'un site.

Quelques-unes des règles sont décrites sous la forme d'expressions régulières. Par exemple, la règle numéro 5 augmente le score des pages qui contiennent dans leur balise `<title>` les termes *home* ou *welcome*. D'autres règles se basent sur la structure des liens, plus particulièrement sur le nombre de liens entrants et/ou sortants d'une page donnée. Par exemple, la règle numéro 9 suppose que plus il y a des liens externes (originaires de pages situées dans d'autres sites) pointant sur une page donnée, plus grande est la probabilité que cette page soit la page d'entrée d'un domaine logique.

Une fois que les scores des pages ont été calculés et mis en ordre décroissant, les k pages les mieux classées, où k est un paramètre de l'algorithme, sont retenues comme pages d'entrée de domaines logiques. L'étape suivante consiste à définir la frontière des domaines logiques ayant comme racine les pages sélectionnées dans l'étape précédente. Ainsi, si le site analysé contient n pages dont k sont choisies comme points d'entrées de domaines logiques, la deuxième étape consiste à affecter les $n - k$ pages restantes à l'un et seulement l'un des domaines logiques ayant comme racine les k pages sélectionnées.

Pour la deuxième étape de l'algorithme, **deux méthodes** ont été proposées. La **première méthode** utilise uniquement l'URL des pages pour définir dans quel domaine logique elles s'insèrent. Définissons le *répertoire hôte* d'une URL comme étant le répertoire dans lequel le fichier associé à l'URL se trouve stocké dans le système de fichiers de la machine d'accueil de l'URL. Par exemple, considérons l'URL `http://www.aaa.com/dirA/dirB/toto.html`. Dans cette URL nous identifions la machine d'accueil `www.aaa.com`, le nom du fichier représentant la page `toto.html` et le *répertoire hôte* `/dirA/dirB/`. L'hypothèse considérée par la première méthode (aussi par la deuxième) est que les répertoires hôtes des URLs des pages d'un domaine logique donné doivent se situer dans le sous-arbre[39] qui a comme racine le répertoire hôte de l'URL de la page d'entrée du domaine logique en question. Autrement dit, le répertoire hôte de l'URL de la page d'entrée d'un domaine logique doit être un préfixe des répertoires hôtes des pages faisant partie de ce même domaine.

Imaginons un domaine logique D_1 ayant comme page d'entrée la page associée à l'URL `http://www.aaa.com/dirA/index.html`. Selon l'hypothèse décrite avant, une page p_i associée à l'URL `http://www.aaa.com/dirA/dirB/pag1.html` serait affectée à D_1. Supposons

[39] La structure sous-jacente à un système de fichiers peut être associée à celle d'un arbre.

un deuxième cas de figure, semblable à celui que nous venons de décrire, mais où en plus existe d'un deuxième domaine logique D_2 ayant comme page d'entrée la page associée à l'URL `http://www.aaa.com/dirA/dirB/index.html`. Dans ce deuxième cas, p_i serait affectée à D_2 et non pas à D_1. Ceci est dû au fait que le répertoire hôte de l'URL de la page d'entrée de D_2 est un préfixe plus long du répertoire hôte de p_i que le répertoire hôte de l'URL de la page d'entrée de D_1. Ainsi, selon les critères que nous venons de citer, les $n - k$ pages du site sont affectées à l'un des k domaines logiques.

De plus, cette première méthode impose certaines restrictions sur les caractéristiques des domaines logiques à découvrir tel que le nombre minimum de pages par domaine. Par exemple, si un domaine découvert ne contient pas le nombre minimum de nœuds il est fusionné avec un autre domaine. Nous n'expliquons pas ici tous les détails de l'algorithme. Soulignons le fait que l'algorithme associé à cette première méthode peut générer des domaines logiques non-connexes dans la mesure où les pages appartenant à un tel domaine ne sont pas nécessairement atteignables à travers une navigation ayant comme point de départ la page d'entrée du domaine. Il semble pourtant raisonnable de supposer que dans un domaine logique toutes les pages puissent être atteignables à partir de sa page d'entrée.

La **deuxième méthode** proposée pour définir les frontières des domaines logiques intègre la supposition que nous venons de citer : les pages d'un même domaine logique doivent non seulement partager le préfixe du répertoire hôte de leur URL, mais elles doivent aussi être accessibles à partir de la page d'entrée du domaine. Ainsi, en plus de l'URL des pages, la structure de liens reliant les pages est donc utilisée.

À l'instar de la première méthode, la deuxième impose certaines restrictions par rapport aux caractéristiques des domaines logiques à découvrir. En dehors du nombre minimum de pages dans un domaine logique, il est défini un rayon maximal autour de la page d'entrée d'un domaine logique (en termes de la longueur des parcours dans le graphe sous-jacent au site analysé) en deçà duquel les pages appartenant au domaine doivent se trouver. Le rayon constitue donc le troisième paramètre — les deux premiers étant le nombre de pages d'entrée à sélectionner au départ (k) et le nombre minimum de pages d'un domaine logique. La première méthode est un cas particulier de la deuxième si l'on considère que (i) deux pages qui ne sont pas reliées par un chemin sont à une distance infinie l'une de l'autre et (ii) le paramètre rayon est positionné à la valeur infinie.

Notons que ni la première méthode ni la deuxième ne garantissent que toutes les pages analysées soient affectées à un domaine logique à la fin. C'est une conséquence des restrictions liées aux préfixes des URLs et au rayon maximal à l'intérieur des domaines logiques. Cela ne constitue pas un problème pour l'application proposée dans [LKV00] qui est celle de l'organisation des résultats d'un moteur de recherche. Les pages n'ayant pas été affectées à un domaine logique pourraient figurer individuellement dans la liste de réponses du moteur de recherche. Il serait pourtant intéressant d'analyser la raison pour laquelle certaines pages ne se retrouvent dans aucun domaine après l'application de l'algorithme. Constitueraient-elles des domaines logiques à part entière ?

L'algorithme a été testé sur trois sites différents : deux sites d'universités et le site du consor-

tium *w3c* (`www.w3c.org`). Les tests n'ont pas abouti à une conclusion puisque les résultats n'ont pas été évalués. Une manière possible d'évaluer la qualité des résultats consisterait à faire analyser les domaines logiques découverts par un certain nombre de personnes parmi lesquelles les concepteurs des sites analysés. D'autre part, même si cette analyse avait été faite, les conclusions ne seraient pas vraiment fiables vu le nombre réduit de sites utilisés dans les tests.

La critique principale qui peut être faite à ce type d'approche est sa dépendance trop importante au type des données utilisées (cf. la règle numéro 2 de la figure 2.8). Pour que l'utilisation de l'algorithme ne soit pas restreinte à un type donné de site, il faudrait adapter/généraliser l'ensemble de règles qui permettent de définir les points d'entrée des domaines. Une deuxième critique est l'utilisation de l'URL des pages pour trouver les différents domaines logiques d'un site. Les hypothèses basées sur les URLs des pages qui sont considérées dans la méthode partent du principe que les fichiers associés aux pages sont organisés hiérarchiquement. L'application de la méthode est donc limité aux sites associés à ce type d'organisation de fichiers (critique déjà évoquée en section 2.2.3.2).

Finalement, les paramètres utilisés par l'algorithme, i.e. le nombre de pages retenues comme pages d'entrée de domaines logiques, le nombre minimum de pages dans un domaine logique et le rayon maximal autour de la page d'entrée d'un domaine logique font que la structure révélée du site — par rapport aux domaines logiques dans lesquels le site est réellement divisé — est imposée, artificielle. Selon le type d'application où les domaines logiques seraient utilisés, l'utilisation de domaines logiques reflétant une structure naturelle pourrait s'avérer fondamentale pour l'efficacité de l'application. Par exemple, l'indexation de sites ou la création d'une carte pour faciliter la navigation d'un utilisateur au sein d'un site (cf. section 3.4.2) sont deux applications pour lesquelles l'utilisation d'une structure reflétant l'organisation naturelle des données est très important.

Après la requête

Dans ce qui suit nous présentons les méthodes qui découvrent les unités logiques d'information après la requête. Tel que nous l'avons vu plus haut, les principaux avantages de ce type de méthode par rapport aux méthodes qui découvrent les ULIs avant la requête sont (i) l'utilisation d'une information fraîche pour construire les index et (ii) le fait de ne pas demander une allocation de ressources en termes d'espace de stockage importante pour être implémentée.

Les méthodes que nous citons dans la suite sont toutes basées sur la découverte de sous-graphes du graphe sous-jacent à l'espace de recherche considéré. Un sous-graphe est considéré comme une réponse pertinente à une requête donnée si les pages associées à ses nœuds couvrent tous les termes présents dans la requête. Il est question donc de répondre à des requêtes conjonctives (composées à l'aide de l'opérateur booléen *and*). Cependant, des adaptations peuvent être facilement conçues afin que le système puisse répondre à des requêtes booléennes plus complexes intégrant des disjonctions. Nous introduisons le concept de sous-graphes minimaux en tant que réponse à des requêtes conjonctives avant d'aborder les méthodes qui utilisent ce concept.

Les sous-graphes minimaux

Les trois méthodes que nous citons dans cette section utilisent un même type d'heuristique pour trouver les unités logiques d'information. L'heuristique consiste à identifier des sous-graphes connexes du graphe sous-jacent à l'espace de recherche considéré par le moteur de recherche[40] qui couvrent l'ensemble des termes d'une requête donnée. Les sous-graphes qui respectent ces deux conditions sont considérés comme des unités logiques d'information pertinentes à la requête en question.

Pour préciser ce que nous entendons par un graphe qui couvre tous les termes de la requête considérons $G = (S, A)$ le graphe, orienté ou non selon les méthodes, sous-jacent à un espace de recherche donné (dans notre contexte les pages d'un site) et $Q = K_1 \wedge \cdots \wedge K_n$ la requête conjonctive composée des termes $K_1, \cdots, K_n$. Définissons D le dictionnaire des termes de l'espace de recherche et $\pi : S \rightarrow \mathcal{P}(D)$ la fonction qui associe à chaque élément de S l'ensemble des termes que la page associée contient[41]. Un sous-graphe $G' = (S', A')$ couvre tous les mots de la requête Q si et seulement si

$$\bigcup_{s \in S'} \pi(s) \supseteq \{K_1, \cdots, K_n\} \tag{2.5}$$

L'idée des méthodes que nous décrivons dans la suite consiste à retourner les sous-graphes pertinents à la requête qui de plus vérifient la condition de minimalité suivante. Cette condition s'exprime par rapport aux nœuds dont le sous-graphe est composé. Reprenons $G' = (S', A')$ comme étant un sous-graphe connexe respectant la condition de couverture citée ci-dessus. L'ensemble de nœuds S' peut être divisé en deux sous-ensembles : le premier est composé de nœuds contenant au moins un terme de la requête, nous le noterons $\overline{S'}$; le deuxième est composé de nœuds ne contenant aucun terme de la requête, nous le noterons $\overline{\overline{S'}}$. Un sous-graphe est minimal si tous les nœuds de $\overline{\overline{S'}}$ sont nécessaires pour maintenir la connexité de G'. Autrement dit, G' n'est pas minimal s'il existe au moins un nœud dans $\overline{\overline{S'}}$ qui peut être supprimé sans que pour autant G' devienne non-connexe.

La différence entre les systèmes qui utilisent les sous-graphes minimaux comme des unités logiques d'information à retourner en réponse à des requêtes se trouve dans la manière dont les sous-graphes correspondant aux ULIs sont découverts et dans la fonction de classement utilisée pour définir les rangs des sous-graphes dans la réponse. Les méthodes basées sur les sous-graphes minimaux que nous allons citer dans la suite diffèrent aussi par l'utilisation du graphe de départ G comme étant orienté ou non.

2.2.3.4 La méthode des chemins conceptuels

Le système *Jumping Spider* est décrit dans [Dyr98] et [Dyr97]. La motivation à la base de la conception de ce système est de permettre une meilleure représentation de l'information véhiculée par une page lorsqu'il s'agit d'évaluer sa pertinence à une requête donnée[42].

[40] Ici, l'application de la découverte des ULIs se trouve dans les moteurs de recherche. L'espace de recherche correspond à l'ensemble des pages indexées par le moteur.

[41] Pour un ensemble E, $\mathcal{P}(E)$ désigne l'ensemble de toutes les parties de E.

[42] Nous parlons de représentation de l'information d'une page plutôt que d'index car dans cette méthode les ULIs sont découvertes après la requête. Or, lorsque l'on parle d'index on fait traditionnellement référence à une représentation construite avant la requête.

L'idée principale est que le ou les concepts qui représentent l'information véhiculée par une page ne peuvent pas être déduits uniquement du contenu de la page. L'hypothèse est qu'un concept peut enjamber plusieurs pages. Les auteurs supposent qu'un concept se traduit par un chemin de pages dans le graphe sous-jacent au site analysé. Un tel chemin est appelé *un chemin conceptuel.* Le système *Jumping Spider* intègre donc une méthode capable d'identifier les chemins conceptuels d'un site, chacun des chemins étant utilisé dans la déduction du(des) concept(s) traité(s) par une page. Bien entendu, un chemin conceptuel peut être considéré comme étant une ULI faisant partie de ce site puisqu'il définit un concept ce qui constitue un volume d'information auto-explicative.

Les chemins conceptuels ont une structure particulière : les concepts qu'ils représentent doivent être raffinés au long des chemins. Autrement dit, un chemin donné doit partir d'un nœud traitant d'un concept de manière générique et se poursuivre vers des nœuds traitant le même concept d'une manière de plus en plus spécifique. On peut aussi considérer que le contenu d'un nœud contextualise le contenu du nœud qui suit. Cependant, dans le Web les liens sont de toutes sortes et, par conséquent, ne mènent pas forcément d'une information plus générique vers une information plus spécifique. Les auteurs proposent donc une analyse des types de liens entre les pages.

La méthode de découverte des chemins commence par une restructuration du graphe sous-jacent au site Web qui constitue l'espace de recherche considéré. La restructuration consiste à ajouter/supprimer des liens entre les nœuds du graphe sous-jacent au site. De cette restructuration résultera un nouveau graphe que nous appellerons dans la suite *graphe modifié.* Les chemins existant dans le graphe modifié correspondent aux chemins conceptuels recherchés. La première étape de la restructuration consiste à classer les liens du graphe original selon trois types :

- lien vers le bas : lien qui pointe vers une page contenant de l'information plus spécifique que l'information contenue dans la page d'où le lien émane ;
- lien de retour : lien qui pointe vers une page contenant de l'information plus générale que l'information contenue dans la page d'où le lien émane ;
- lien de côté : lien qui pointe vers une page dont le repértoire hôte[43] de l'URL n'a aucun lien de parenté avec le repértoire hôte de l'URL de la page d'où le lien émane. Deux répertoires Rep_a et Rep_b n'ont pas de lien de parenté si Rep_a n'est ni un préfixe (répertoire ancêtre) ni un suffixe (sous-répertoire) de Rep_b. Un lien vers une page hébergée par un autre serveur HTTP est aussi un lien de côté.

Afin de savoir si un lien pointe sur une page contenant de l'information plus spécifique ou plus générique que celle contenue dans la page d'où le lien émane (cf. définition de lien de retour et lien vers le bas) les répertoires hôtes des URLs sont analysés. L'hypothèse considérée est que l'information stockée dans un répertoire donné, disons $/Rep_x/Rep_y/Rep_z/$ est plus spécifique que l'information stockée dans ses répertoires ancêtres, e.g. $/Rep_x/$ et $/Rep_x/Rep_y/$. Comme nous l'avons déjà souligné dans les critiques des méthodes citées dans les section 2.2.3.2 et section 2.2.3.3, cette hypothèse ne peut être vérifiée que dans des sites où (i) l'organisation des fichiers associés aux pages est hiérarchique et (ii) la profondeur

[43] Regarder la définition dans la section 2.2.3.3.

relative du fichier dans la hiérarchie du système de fichiers est corrélée avec le degré de spécificité/généricité de l'information qu'il contient.

La classification que nous venons de d'évoquer ne permet pas de classer les liens vers des pages situées dans le même répertoire que les pages d'où les liens émanent. Les liens de ce type sont classés de la manière suivante : (i) si le lien émane d'une page dont le nom de fichier est du genre `index.htm(l)` le lien est classé *lien vers le bas* ; (ii) si, à l'opposé du premier cas, le lien pointe sur une page dont le nom de fichier est `index.htm(l)` le lien est classé *lien de retour* ; (iii) pour les liens entre deux pages stockées dans un même répertoire qui ne rentrent pas dans les deux premiers cas, si la page pointée par le lien en train d'être analysé est aussi pointée par un deuxième lien déjà classé *lien vers le bas* ou *lien de côté*, le lien en train d'être analysé est classé *lien vers le bas*, dans le cas contraire, le lien est classé *lien de côté*.

Après avoir classé tous les liens du graphe de départ, le processus de restructuration s'accomplit avec la construction du graphe que nous avons appelé plus haut graphe modifié dans lequel les chemins conceptuels sont identifiés. La construction se déroule en deux étapes :

1. tous les liens du graphe original exceptés ceux classés *liens de retour* sont reproduits dans le nouveau graphe ;
2. un lien est ajouté au nouveau graphe entre le premier et le dernier nœud d'un chemin composé d'une séquence de liens *vers le bas* suivie éventuellement d'un lien de côté.

Notons que les pages représentées par les nœuds du nouveau graphe peuvent appartenir à plusieurs chemins conceptuels à la fois. Par rapport à l'indexation, les pages sont indexées et non pas les chemins conceptuels auxquels elles appartiennent. Les pages sont indexées uniquement d'après leur contenu et la structure du graphe modifiée est également sauvegardée. Les chemins conceptuels des pages sont découverts pour chaque requête soumise au moteur de recherche. Bien-entendu, tous les chemins conceptuels existant dans le graphe modifié ne sont pas découverts à chaque requête soumise, seuls ceux qui intègrent certaines pages identifiées par une première étape de la fonction de correspondance — que nous résumons plus bas — sont découverts.

En ce qui concerne la requête de l'utilisateur, elle possède une structure particulière. Puisque les chemins conceptuels possèdent une structure reflétant une généralisation/spécialisation de concepts, les requêtes traitées par le système *Jumping Spider* présentent ce même type de structure. Observons que cela implique une supposition selon laquelle l'utilisateur est capable de traduire son besoin d'information dans une requête ayant une telle structure. Par exemple, la requête *ordinateur* $\rightarrow$ *réseau* traduit le besoin d'information sur les réseaux d'ordinateurs ; *ordinateur* étant le terme plus général de la requête qui contextualise son terme plus spécifique *réseau*[44]. Nous pouvons citer comme inconvénient de ce type de requête l'effort que l'utilisateur doit fournir afin de traduire son besoin d'information sous le

[44]Les relations de spécificité et généralité entre des termes peuvent être différentes selon le contexte où les termes sont considérés. Dans le contexte du besoin d'information *réseaux d'ordinateurs* le terme *ordinateur* peut être considéré comme un terme plus général que le terme *réseau* puisqu'il contextualise ce dernier terme.

format imposé.

La fonction de correspondance du moteur de recherche utilise le graphe modifié pour décider si une page est pertinente ou non à une requête. Ci-dessous nous résumons l'algorithme proposé pour la fonction de correspondance. Considérons que (i) la table *Index(concept, url)* spécifie les concepts se trouvant dans chaque page du site analysé et (ii) la table *Reachable(from, to)* contient la structure du graphe modifié.

- Entrées : la table *Index(concept, url)*, la table *Reachable(from, to)* et une liste de mots-clés $k_1, \cdots, k_n$ représentant la requête $Q = k_1, \cdots, k_n$.

- Soit $U_1, \cdots, U_n$, *active*, *canreach* et *result* des ensembles vides d'URLs et z une URL.
- /* Associer à chaque terme de la requête k_i un ensemble d'URLs U_i */
- U_i := SELECT *url* FROM *Index* WHERE concept = k_1
- ...
- U_n := SELECT *url* FROM *Index* WHERE concept = k_n
- /* Tester chaque page candidate, i.e. celles contenant le concept k_n */
- Pour chaque $z \in U_n$ faire
  - *active*:=z
  - /* Itérer sur les $n-1$ premiers termes de la requête */
  - for i:=$n-1$ to 1 faire
    - /* Pour chaque $u \in$ *active* déterminer tous les nœuds pointant sur u */
    - *canreach* := SELECT *from* FROM *Reachable* WHERE *to* IN *active*
    - *active* := *canreach* $\cap$ U_i
  - /* Si *active* $\neq \emptyset$ alors il existe un chemin conceptuel */
  - Si *active* $\neq \emptyset$ alors *result* := *result* $\cup$ z
- Renvoyer *result*

Remarquons que l'algorithme analyse les termes de la requête un par un mais en ordre inverse, i.e. l'analyse débute par le dernier terme k_n, celui-ci représentant le concept le plus spécifique de la requête. Par exemple, reprenons le besoin d'information utilisé comme exemple dans la section 2.2.1 et décrit par : « notes de cours sur les arbres dans un cours de structure de données. » Ce besoin d'information se traduit dans la requête : *structures de données* $\rightarrow$ *notes de cours* $\rightarrow$ *arbres*. *Structures de données* est le terme le plus générique et *arbres* est le terme le plus spécifique de cette requête.

Une réponse classée pertinente à une requête donnée correspond à un chemin dans le graphe modifié menant d'un nœud contenant le terme le plus général de la requête à un autre nœud contenant le terme le plus spécifique de la requête (cf. figure 2.9). Ce chemin peut être considéré comme étant un sous-graphe minimal du graphe modifié. Une particularité des sous-graphes réponses trouvés par l'algorithme est que tous ses nœuds contiennent au moins un terme de la requête, autrement dit il n'y a pas de nœuds ne contenant aucun terme de la requête mais qui sont présents juste pour maintenir la connexité du graphe. Cette particularité est une conséquence de la deuxième étape de la construction du graphe qui ajoute un nouveau lien entre toute paire de pages où l'une des pages est censée traiter d'un concept d'une forme plus générale/spécifique que l'autre page.

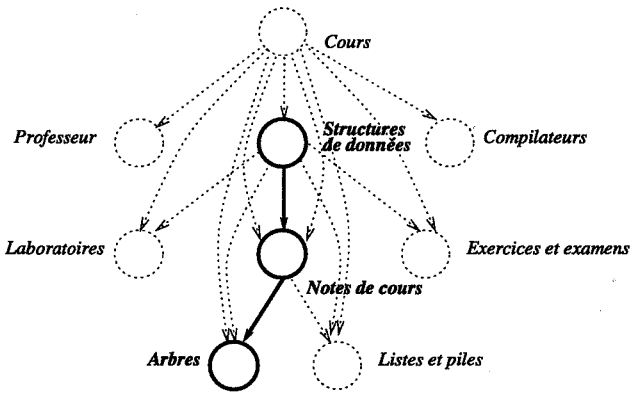

FIG. 2.9: Partie du graphe du Web restructuré. En gras, graphe associé au chemin conceptuel pertinent à la requête *structures de données* → *notes de cours* → *arbres*

Notons aussi que les pages et non pas les chemins conceptuels sont rendus en réponse aux requêtes, la page contenant le terme le plus spécifique de la requête étant retournée. Par ailleurs, malgré le fait que l'algorithme ne prévoit pas la situation où plus d'un terme de la requête, voire tous, se retrouveraient dans une même page, il peut cependant être facilement modifié de sorte que cette situation soit prise en compte.

La critique que nous pouvons faire est l'utilisation de la structure codée dans l'URL des pages pour (i) découvrir les unités logiques d'information, i.e. les chemins conceptuels et (ii) déduire une relation de spécificité/généralité entre l'information contenue dans les pages d'après le type d'organisation des fichiers associés aux pages dans le système de fichiers. Certes il s'agit d'une heuristique simple et facile à implémenter cependant il reste à démontrer son efficacité. Aucune évaluation n'est proposée dans les articles dont ce système a fait l'objet.

2.2.3.5 La méthode basée sur le problème de Steiner

Dans [LCVA01] on propose un moteur de recherche qui rend des unités logiques d'information qu'il découvre et classe. L'implémentation des qualités imposées à ce classement nécessite une méthode de découverte de sous-graphes minimaux que les auteurs décrivent aussi dans leur article. Considérons le sous-graphe minimal $G' = (S', A')$ pertinent à une requête donnée. Dans [LCVA01], le classement de G' est défini par son coût. Le coût de G' est défini par la somme des poids des arêtes de l'arbre couvrant minimal des nœuds de $\overline{S'}$. Rappelons que $\overline{S'}$ est le sous-ensemble de S' dont les nœuds correspondent aux pages contenant des termes de la requête. L'hypothèse considérée dans [LCVA01] est que plus le coût d'un sous-graphe réponse est réduit, plus l'ULI représentée par le sous-graphe est pertinente.

Le problème qui consiste à trouver dans un graphe l'arbre qui couvre un sous-ensemble de ses nœuds et dont le coût est minimal — le coût étant calculé par la somme des poids des arêtes — a déjà été défini et étudié dans la littérature sous le nom de « *l'arbre de groupes de Steiner.* » Considérons $R = \{R_1, \cdots, R_m\}$ où R_i est un sous-ensemble des sommets du

graphe $G = (S, A)$[45]. Le problème de la découverte de l'arbre de groupes de Steiner consiste à trouver l'un des sous-graphes connexes à coût minimal G' de G tel qu'au moins un élément de chaque ensemble R_i ($R_i \in R$) appartienne à l'ensemble des sommets de G'. Ce problème est connu comme étant NP-complet. Il est démontré que si un tel sous-graphe de G existe il constitue un arbre.

Si nous considérons une requête conjonctive $Q = K_1 \wedge K_2 \wedge \cdots \wedge K_m$ et que R_i représente l'ensemble de pages contenant le terme K_i nous pouvons modéliser le problème de la découverte d'ULIs en utilisant le problème de Steiner et pour le résoudre il suffit d'utiliser l'une des nombreuses heuristiques proposées jusqu'à ce jour dans la littérature. Cependant la méthode de découverte d'ULIs envisagée par Li dans [LCVA01] doit en plus satisfaire les conditions suivantes :

- les k plus pertinentes unités logiques d'information doivent être découvertes et non pas seulement l'unité la plus pertinente[46] ;
- le calcul de la *n-ième* réponse doit réutiliser les calculs effectués pour la découverte des *(n-1) premières* réponses ;
- la méthode ne doit pas nécessiter d'analyser l'espace de recherche complet, i.e. énumérer toutes les pages considérées par le moteur de recherche en question, afin de découvrir des ULIs pertinentes à une requête donnée.

Les deux dernières conditions sont liées au fait que les méthodes qui découvrent les ULIs après la requête ont l'inconvénient de demander un temps de réponse assez long et qui peut s'avérer prohibitif vis-à-vis des utilisateurs. Ces conditions se justifient donc par le souhait de diminuer le temps de réponse du moteur de recherche.

Cependant, d'après les auteurs, les méthodes proposées jusqu'à ce jour pour le problème des arbres de groupe de Steiner ne satisfont aucune des trois conditions. Toutes calculent seulement l'arbre couvrant minimal et non pas les k-premiers en ordre croissant de coût. En plus, une énumération de tous les nœuds du graphe de l'espace de recherche considéré par le moteur de recherche est nécessaire pour l'application des méthodes/heuristiques. Dans [LCVA01] Li propose une nouvelle heuristique pour le problème des arbres de groupe de Steiner qui satisfait les trois conditions citées plus haut.

L'heuristique peut être résumée de la manière suivante. Considérons la requête conjonctive $Q = K_1 \wedge K_2 \wedge \cdots \wedge K_m$. À l'aide d'un moteur traditionnel on peut expliciter les ensembles $R_1, R_2, \cdots, R_m$. L'algorithme commence son exécution par l'analyse d'un graphe, que nous allons appeler ici G^*. Au départ G^* est un graphe sans arêtes dont les sommets sont les sommets de l'ensemble $\bigcup_{i=1\cdots m} R_i$, par conséquent, G^* est composé au départ d'au plus $\sum_{i=1}^{m} |R_i|$ composantes connexes. L'heuristique consiste à modifier itérativement G^* en y ajoutant des arêtes et des nœuds.

La figure 2.10(a) illustre G^* au début de l'exécution de l'algorithme pour la requête $Q = K_1 \wedge K_2 \wedge K_3$. Les nœuds et arêtes de G^* y sont dessinés en trait continu. Les arêtes

[45] Insistons sur le fait que $R_i \subseteq S$, ou encore $R_i \in \mathcal{P}(S)$.

[46] L'utilisateur est normalement intéressé par les k réponses les plus pertinentes à sa requête et pas seulement par la meilleure réponse.

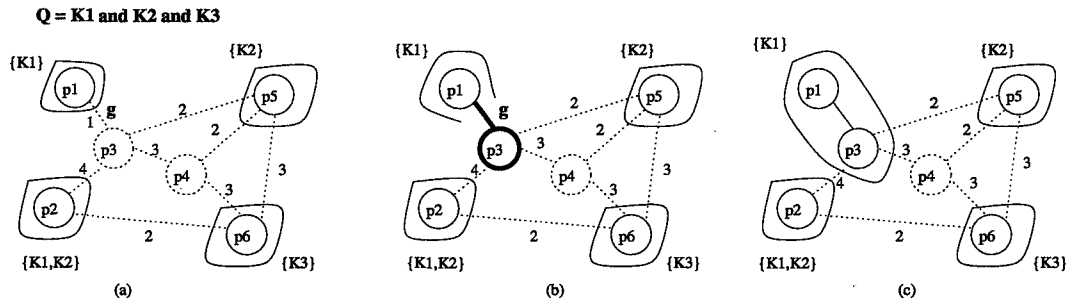

FIG. 2.10: Sous-ensemble de règles utilisées pour classer les pages candidates à devenir pages d'entrée des domaines logiques.

et les nœuds en pointillé ne font pas partie de G^*, ils représentent des arêtes et des nœuds présents dans G, le graphe original sous-jacent à l'espace de recherche considéré. Les arêtes dessinées en pointillé seront appelés dans la suite *arêtes inexploitées.*

Au début d'une itération de l'heuristique chacune des composantes connexes de G^* est considérée comme **un graphe candidat** à devenir une réponse de l'algorithme, c.-à-d. un arbre minimal couvrant tous les termes de la requête. À chaque itération, l'heuristique choisi parmi les arêtes inexploitées de G ayant une attache dans G^* celle associée au poids le plus petit. En plus cette arête doit relier un des nœuds d'un des graphes candidats soit à un nœud d'un autre graphe candidat soit à un nœud qui ne fait partie d'aucun graphe candidat — ce dernier type de nœud sera appelé dans la suite *nœud libre.* Par exemple, dans la figure 2.10(a) il existe quatre graphes candidats et le nœud p_3 est un *nœud libre.*

Si l'arête choisie est attachée à un *nœud libre*, cette arête et ce nœud sont intégrés au graphe candidat aussi attaché par l'arête choisie. Si l'arête choisie relie deux nœuds appartenant à deux graphes candidats différents, ces deux graphes sont fusionnés en un seul. Une fois que l'intégration est faite, l'algorithme vérifie si le graphe candidat concerné par l'intégration, appelons-le G_c, couvre tous les mots de la requête (cf. équation 2.5). Si tel est le cas, l'arbre couvrant minimal du sous-graphe de G composé des nœuds de G_c contenant au moins un terme de la requête est calculé et retourné en réponse. Si le nombre demandé de réponses — paramètre de l'algorithme — est atteint l'algorithme finit son exécution, dans le cas contraire, l'algorithme réitère en choisissant une nouvelle arête parmi celles encore inexploitées.

Dans la figure 2.10, nous illustrons une itération de l'algorithme. Au début de l'itération (figure 2.10(a)) il existe 4 graphes candidats (nœuds encerclés) et l'arête choisie pour être exploitée est l'arête g (dessinée en gras) car parmi toutes les arêtes encore inexploitées (en pointillé) qui répondent aux critères de sélection évoqués plus haut, l'arête g est celle associée au poids le plus petit. Le graphe candidat contenant le nœud p_1 est celui relié par l'arête g et, par conséquent, il intègre l'arête g ainsi que le nœud libre associé à g, en l'occurrence p_3 (l'arête et le nœud intégrés sont dessinés en gras dans la figure 2.10(b)). Le candidat s'agrandit en devenant un graphe contenant deux nœuds et une arête (figure 2.10(c)). Comme le graphe candidat qui vient d'être agrandi ne constitue pas une réponse à la requête (il ne contient que le terme K_1), une nouvelle arête est choisie parmi celles encore inexploitées et

le processus d'intégration que nous venons de décrire est réitéré.

Au fur et à mesure que les k arbres de groupes de Steiner sont découverts ils sont retournés à l'utilisateur. Puisque l'algorithme découvre les arbres en ordre croissant de coût, les ULIs correspondantes sont retournées à l'utilisateur en ordre décroissant de pertinence. Notons que cette caractéristique de l'algorithme est compatible avec l'une des trois conditions que nous avons énumérées plus haut.

En ce qui concerne le format des résultats de l'algorithme, en réalité ce n'est pas une liste de graphes — correspondant aux arbres de groupe de Steiner — qui est renvoyée à l'utilisateur en réponse à sa requête. Considérons un graphe $G' = (S', A')$ correspondant à l'une des réponses retrouvées par l'algorithme. Nous avons souligné plus haut lors de la définition des sous-graphes minimaux que parmi les nœuds de S' nous pouvons identifier des nœuds contenant au moins un terme de la requête, ces nœuds composent l'ensemble que nous avons noté $\overline{S'}$. L'unité logique d'information retournée par l'algorithme est constituée uniquement des pages associées aux nœuds de l'ensemble $\overline{S'}$. En realité, la structure du graphe n'est prise en compte que pour la découverte des ULIs et pour le calcul du coût des sous-graphes qui leur sont associés, ce qui, d'après l'hypothèse posée dans l'algorithme, permet d'estimer la pertinence des ULIs. Cependant, comme nous l'avons souligné dans la section 2.2.2.1, il se peut qu'une réponse composée d'un ensemble de pages ne constitue pas le format le plus adéquat pour rendre une réponse aux utilisateurs. Une manière alternative d'afficher un tel type de réponse consiste à afficher l'arbre de groupe de Steiner ayant généré la réponse. Encore faut-il vérifier si un tel affichage serait bénéfique à l'interprétation des résultats par l'utilisateur. Dans l'article cette question n'est pas évoquée.

Des tests ont été menés sur des graphes synthétiques générés au hasard ainsi que sur des graphes représentant des données extraites du Web. L'objectif de ces tests a été d'évaluer l'efficacité de l'algorithme de découverte d'ULIs, en l'occurrence de la découverte des arbres de groupe de Steiner, et non pas d'évaluer l'efficacité du moteur de recherche qui utilise les ULIs comme unité d'information. Comme nous l'avons dit plus haut le problème de la découverte des arbres de groupe de Steiner est NP-complet. L'heuristique que nous venons d'évoquer essaye de résoudre ce problème[47]. Évidemment, les résultats fournis par l'heuristique ne sont pas optimaux et les tests ont permis d'estimer la qualité des résultats fournis par rapport aux résultats optimaux qui ont été identifiés à travers une recherche exhaustive. La qualité a été mesurée en comparant pour chaque réponse le rang estimé par l'heuristique et celui calculé par l'analyse exhaustive. Rappelons que la réponse associée à un sous-graphe minimal est constituée seulement des nœuds qui contiennent les termes de la requête. Ainsi, l'analyse exhaustive peut trouver une réponse donnée à travers un sous-graphe minimal et l'heuristique que nous décrivons ici peut trouver cette même réponse à travers un autre sous-graphe minimal. Comme il peut s'agir de deux sous-graphe différents, les coûts de la réponse qu'ils représentent peuvent être différent, d'où la différence au niveau du rang estimé.

Certes ce type d'évaluation permet de mesurer la sub-optimalité de l'heuristique mais cela

[47]En fait, l'heuristique proposée résout un problème encore plus complexe que celui de l'arbre de groupes de Steiner. L'heuristique trouve non seulement l'arbre couvrant minimal (résolution du problème de base de Steiner) mais les k premiers arbres couvrants minimaux.

ne permet pas de mesurer l'efficacité d'un moteur de recherche basé sur la notion d'ULIs. Les hypothèses faites dans ce travail sont (i) tout ensemble de pages reliées par la structure du graphe sous-jacent à l'espace de recherche considéré et couvrant tous les mots de la requête constitue une unité logique d'information pertinente à la requête posée et (ii) plus le coût du graphe associé à une ULI est petit plus l'ULI est pertinente. Si ces hypothèses s'avèrent être toujours vraies l'efficacité de l'algorithme peut être considérée comme étant de 100% du fait qu'il produit uniquement des réponses dont les pages couvrent tous les mots de la requête. Cependant, il se peut que ces hypothèses ne s'avèrent pas toujours vraies. Il serait donc intéressant de pouvoir estimer la validité de ces hypothèses à travers l'analyse de la réelle pertinence des ULIs renvoyées par le moteur en réponse aux requêtes soumises.

Finalement, le facteur qui possède vraisemblablement la plus grande influence dans la découverte des unités logiques d'information dans l'algorithme a été relativement ignoré par les auteurs. Ce facteur est les poids associés aux liens. Ce sont les liens qui font que deux pages peuvent faire éventuellement partie d'une ULI pertinente à une requête donnée. Plus le poids associé au lien entre deux nœuds est élevé, pire est le rang d'une éventuelle ULI incluant les pages associées aux deux nœuds et, par conséquent, moins pertinente est l'ULI. De ce fait la méthode de pondération des liens a une forte influence dans le classement des ULIs. Paradoxalement, dans les tests effectués sur les données extraites du Web, tous les liens ont été associés à un même poids de valeur égale à 1. Ce choix n'a pas été justifié.

2.2.3.6 La méthode des liens de navigation

Comme la méthode évoquée dans la dernière section, la recherche de sous-graphes minimaux contenant les mots clés saisis dans la requête est utilisée par la méhode que nous résumons dans la suite. La différence entre les deux se trouve dans la manière dont le graphe de pages est construit et la méthode de classement des sous-graphes récupérés.

L'algorithme peut se résumer en trois étapes : (i) la construction du graphe sur lequel l'identification d'ULIs se fera ; (ii) l'identification des ULIs ; (iii) le classement des ULIs en ordre décroissant de pertinence à la requête formulée. On peut considérer que les deux premières étapes se déroulent en parallèle.

La première étape utilise un moteur de recherche traditionnel pour récuperer l'ensemble des pages autour desquelles le graphe sera construit. À l'instar des deux derniers systèmes décrits, le système que nous présentons dans cette section s'intéresse aux requêtes conjonctives. Une conjonction est une proposition logique dont les composants sont regroupés par l'opérateur *and*. Une conjonction est vraie si et seulement si tous ses composants sont vrais. Afin de récupérer un premier ensemble de pages, le système transforme la requête conjonctive en une requête disjonctive et l'envoie à un moteur de recherche traditionnel qui renvoie une liste d'URLs en réponse.

Ensuite, les liens émanant de chaque page associée aux URL retournées sont analysés. L'idée ici est d'identifier parmi tous les liens qui partent de chaque page, ceux qui expriment une relation de complémentarité avec la(es) page(s) pointée(s). Ces liens sont appelés *liens de navigation* par les auteurs. Plus bas nous évoquons la manière dont les liens de ce type sont identifiés. Après, les liens de type navigation sont traversés et les pages pointées sont réunies

aux pages d'où les liens émanent. L'opération précédente génère un ensemble de graphes orientés contenant éventuellement des cycles. Chaque graphe est ensuite analysé de sorte à vérifier si l'ensemble de ses pages couvre l'ensemble de mots clés saisis dans la requête. Un graphe dont l'ensemble de pages contient tous les mots de la requête constitue une réponse pertinente à la requête posée. Si le nombre de réponses trouvées est inférieur à un nombre prédéfini de réponses fourni en paramètre de l'algorithme, la méthode est réitérée, i.e. les liens émanant des pages récemment intégrées aux graphes qui ne constituent pas encore des réponses sont suivis ; les pages pointées sont récupérées et les graphes générés sont analysés. La partie de l'algorithme qui décrit ces deux premières étapes est résumée ci-dessous :

- Entrées : n et Q où n correspond au nombre de réponses souhaitées et Q est une requête conjonctive.

- **1.** Traduire $Q = k_1$ *and* k_2 *and* k_3 sous la forme $Q' = k_1$ *or* k_2 *or* k_3 et soumettre Q' à un moteur traditionnel. Chaque page retournée est associée à un graphe candidat singleton.
- **2.** Seuls les *liens de navigation* des pages sont traversés; les pages pointées ainsi que les liens sont intégrés aux graphes candidats contenant les pages d'où les liens traversés émanent.
- **3.** Les graphes générés par l'étape précédente sont analysés de sorte à identifier ceux qui couvrent éventuellement tous les mots de la requête. S'il en existe, les graphes sont placés dans l'ensemble *Resp*. Si $|Resp| < n$ alors retourner à l'étape 2.

Remarquons que les graphes réponses générés par l'algorithme sont orientés et constituent des chemins (deux exemples sont illustrés dans la figure 2.11). Souvent, deux ou plusieurs graphes réponses partagent l'ensemble de pages contenant les termes de la requête. La seule différence entre eux sont les nœuds satellites qui ne contiennent pas de termes de la requête et dont la seule fonction dans le graphe est de maintenir sa connexité. Dans cette situation, seul le graphe le mieux classé est retenu dans la réponse finale. Plus bas nous décrivons l'algorithme de classement proposé dans la méthode.

L'identification des liens de type *lien de navigation* exécutée dans l'étape 2 s'appuie sur les URLs des pages reliées. Une analyse est faite sur la structure de répertoire codée dans les URLs des pages reliées pour décider si le lien exprime une relation de complémentarité ou non. Cette analyse se sert d'un sous-ensemble d'hypothèses décrites dans [MT99], un travail précédent des mêmes auteurs déjà mentionné plus haut (cf. section 2.2.3.2). Ainsi, seuls les liens de navigation identifiés à l'aide de ces hypothèses sont traversés lors de la découverte des ULIs. Il est important de remarquer que dans l'étape 3 de l'algorithme le fait de ne pas suivre tous les liens partant d'une page mais seuls ceux de type *lien de navigation* réduit considérablement le temps consacré à la découverte des ULIs.

Une fois que les sous-graphes pertinents à la requête ont été trouvés, il faut établir un classement afin de les retourner à l'utilisateur en ordre décroissant de pertinence. Le classement d'un sous-graphe est défini en fonction : (i) du nombre de pages qu'il contient ; (ii) de la *localité* des termes de la requête entre ses pages et (iii) de la *localité* des termes de la requête dans les pages.

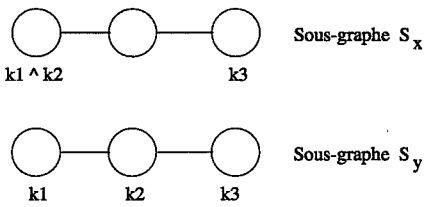

FIG. 2.11: Distribution des termes d'une requête entre les pages de deux graphes minimaux.

La localité d'un terme correspond à la région d'un texte où le terme est censé posséder une certaine influence. Plus les localités de deux termes se recouvrent dans une page, plus grande est la probabilité que ces termes aient une relation entre eux. La prise en compte de la localité des termes peut donc être utile pour distinguer le cas où deux termes ont une relation entre eux du cas où les deux termes n'ont pas de relation.

L'utilisation de la localité des termes de la requête *dans* une page est inspirée du fait que la présence de deux termes dans une même page n'indique pas forcément que les deux termes ont une relation entre eux. Une indication d'une éventuelle relation entre les deux termes dans un texte est le nombre d'autres termes qui les sépare[48]. Cependant, la fonction utilisée pour mesurer le recouvrement des localités de deux termes est plus complexe que celle basée sur la distance entre les deux termes : elle est basée sur la densité d'apparition d'un terme dans un texte et est décrite dans [KSN97].

Pour ce qui est de l'influence de la localité des termes *entre* les pages sur le rang d'un sous-graphe dans la liste de réponses, plus les termes de la requête se retrouvent ensemble dans une même page, plus le rang attribué au graphe est élevé. Imaginons les graphes réponses S_x et S_y pertinents à la requête $Q = k1$ *and* $k2$ *and* $k3$. Ces graphes sont illustrés dans la figure 2.11. Selon la fonction de classement proposée, le graphe S_x est mieux classé que le graphe S_y puisque les termes de la requête *entre* les pages de S_x sont moins dispersés qu'entre les pages de S_y.

Un moteur de recherche a été implémenté pour tester la méthode décrite ici. Des tests ont été conduits mais les résultats n'ont pas été évalués. Des exemples de couples requêtes-/réponses trouvées par le moteur ont montré que dans certaines situations une information pertinente qui ne serait pas retrouvée par un moteur de recherche traditionnel peut être retrouvée grâce à l'utilisation du concept d'ULI. L'une des conclusions tirées des résultats suggère que l'utilisation d'ULIs est surtout utile pour répondre à des requêtes orientées *taux de rappel*, c.-à-d. plus le nombre de documents pertinents présents dans la réponse est grand mieux c'est [49]. Nous ne sommes pas complètement d'accord sur cette conclusion par le fait

[48] Plusieurs moteurs de recherche permettent de spécifier le nombre maximum de termes pouvant exister entre deux termes de la requête dans une page pour qu'elle soit jugée pertinente.

[49] Le taux de rappel, en anglais *recall*, est une mesure traditionnelle pour évaluer l'efficacité d'un système de recherche d'information. Par rapport à la réponse renvoyée à une requête donnée, le taux de rappel est le pourcentage de documents connus comme étant pertinents qui sont présents dans la réponse. Notons que l'espacement entre les documents pertinents dans la liste de réponses n'est pas prise en compte.

qu'elle présuppose que pour la plupart de requêtes il y a toujours de nombreuses pages pertinentes qui sont en réalité des documents à part entière. Or, à travers la découverte des ULIs on cible surtout la récupération des pages qui ne constituent pas des documents à part entière. Finalement, la question de la présentation des ULIs pertinentes à l'utilisateur n'a pas été évoquée.

2.2.4 Comparaison des méthodes

Dans la prochaine page nous proposons un tableau comparatif des méthodes que nous avons présentées dans les sections précédentes. Nous y récapitulons les différents aspects qui permettent de comparer les méthodes. Nous avons aussi inclus dans le tableau la méthode que nous proposons dans le prochain chapitre afin de la situer par rapport aux autres. Précisons les légendes utilisées dans le tableau :

- **Méth. I** : la méthode des coupures d'un graphe (section 2.2.3.1)
- **Méth. II** : la méthode des chemins d'entrée d'une page (section 2.2.3.2)
- **Méth. III** : la méthode de l'identification des sous-domaines logiques dans un site Web (section 2.2.3.3)
- **Méth. IV** : la méthode des chemins conceptuels (section 2.2.3.4)
- **Méth. V** : la méthode basée sur le problème de Steiner (section 2.2.3.5)
- **Méth. VI** : la méthode des liens de navigation (section 2.2.3.6)

En analysant le tableau nous pouvons dégager quelques tendances sur la manière d'aborder la problématique de la découverte d'unités logiques d'information. Premièrement, à l'exception des méthodes IV et V, toutes les méthodes supposent qu'une unité logique d'information n'enjambe pas deux sites différents[50]. Bien que la méthode III utilise de l'information non locale (les liens qui pointent vers les pages du site en train d'être analysé) c'est uniquement pour découvrir les ULIs, les ULIs étant composées uniquement de pages internes au site. Ainsi, le fait que de l'information non locale soit utilisée pour découvrir une ULI n'implique pas forcément que l'ULI contienne de l'information non locale.

La méthode IV permet que dans une ULI il y ait au plus une page située dans un site différent de celui des autres. Ceci est dû à la définition de chemin conceptuel selon laquelle un tel chemin peut finir par un lien de type *lien de côté* (lien entre deux pages situées dans deux sites différents).

Pour ce qui concerne la méthode V, bien que l'utilisation de d'information non locale soit suggérée dans l'article, seule l'information locale a été utilisée dans les expérimentations conduites. Ce choix a été justifié par les auteurs comme étant une question de faisabilité en termes de coût de calcul.

Nous croyons que cette tendance générale d'utilisation de l'information locale aux dépens de l'information non locale est due à deux raisons principales : (i) alors qu'il semble certain que l'information complémentaire/contextuelle de l'information d'une page donnée soit disponible dans d'autres pages du même site, il n'est pas certain que cette information soit aussi

[50] Ces méthodes considèrent un site comme étant un ensemble de pages dont les URLs partagent le nom de la machine hôte et le numéro de port.

TAB. 2.1: Tableau comparatif des méthodes de découverte d'ULIs

	Méth. I	Méth. II	Méth. III	Méth. IV	Méth. V	Méth. VI	Méth. proposée
La granularité de la réponse	ensemble de pages	une page	N/A	une page	ensemble de pages	ensemble de pages	une page
Le nombre d'unités logiques par page	plusieurs	une	une	plusieurs	plusieurs	plusieurs	une ou plusieurs
Le moment de découverte du graphe	avant	avant	avant	avant	après	après	avant
Le moment de découverte des ULIs	avant	avant	avant	après	après	après	avant
Le type d'information utilisée pour la découverte des ULIs	contenu + liens	liens + URL	contenu + liens + URL	contenu + liens + URL	contenu + liens	contenu + liens + URL	contenu + liens
La prise en compte de l'hétérogénéité des liens	non	oui	oui	oui	non	oui	non
La localité de l'information composant une ULI	locale	locale	locale	non locale	non locale	locale	locale

disponible à l'extérieur du site ; (ii) l'exploitation de l'information non locale est beaucoup plus coûteuse que l'exploitation de l'information locale.

Pour ce qui concerne la granularité de la réponse renvoyée par les moteurs de recherche bénéficiant des ULIs, la plupart des méthodes suggèrent le renvoi de l'ULI en réponse aux requêtes formulées. En effet, il est plus logique de renvoyer une ULI plutôt que l'une des pages composant l'ULI si l'on considère qu'une ULI représente un *document* hypertexte et que la tâche d'un moteur de recherche est de retourner à l'utilisateur des *documents* pertinents à la requête. Cependant, comme nous l'avons signalé dans la section 2.2.2.1, une réponse multi-pages peut être préjudiciable à la compréhension des résultats par l'utilisateur. Aucune des méthodes adoptant un tel format de réponses n'a évoqué ce problème ni, par conséquent, a proposé une visualisation particulière des résultats.

Par rapport aux aspects concernant le nombre d'unités logiques par page et la prise en compte de l'hétérogénéité des liens la tendance semble être claire. D'une part, il semble presque logique qu'une page puisse appartenir à plus d'une unité logique d'information. Ceci est dû au style de rédaction dans l'hypertexte, résultat de la richesse d'associations qui peuvent s'établir entre les informations contenues dans chaque nœud. Les pistes associatives d'un hypertexte évoquées par Bush [Bus45] qui peuvent se croiser sur n'importe quel nœud, suggérant ainsi qu'un nœud peut appartenir à plusieurs documents.

Pour ce qui est de l'hétérogénéité des liens, toutes les méthodes en tiennent compte. Les méthodes font appel à trois types de données afin d'établir une distinction entre les liens qui expriment une relation de complémentarité et sont ainsi censés relier pages appartenant à une même ULI et ceux qui expriment d'autres types de relation : le contenu et les URLs des pages connectés par un lien ou une séquence de liens. À l'aide d'heuristiques basées sur ces deux types de données l'hétérogénéité des liens est prise en compte.

Dans la méthode V les liens du graphe sous-jacent à l'espace de recherche considéré sont associés à des poids. Cette pondération permet de distinguer les liens. Par exemple, nous pouvons imaginer que plus un lien exprime une relation de complémentarité entre deux pages, plus petit doit être son poids (idée implicite dans la méthode). Cependant, dans la description de la méthode, la manière de calculer les poids n'a pas été évoquée et dans les expérimentations citées un même poids a été attribué à tous les liens.

Quant au moment de la découverte du graphe des pages sur lequel le processus de découverte d'ULIs s'exécute et la découverte des ULIs elles-mêmes, les approches proposées se divisent d'une manière équitable entre deux possibilités : découvrir le graphe et les ULIs avant ou après la requête. La seule remarque à faire sur ce point concerne la méthode IV qui adopte une approche mixte. Cette méthode découvre le graphe avant la requête et les ULIs après la requête. De ce fait, elle semble perdre les avantages des approches non-mixtes, à savoir la garantie d'un temps de réponse court pour ce qui est des approches de type *avant la requête* et la fraîcheur de l'information manipulée pour ce qui est des approches de type *après la requête.*

Finalement, abordons le dernier aspect qui concerne le type d'information utilisé par les méthodes afin de découvrir les ULIs. Ce point permettra aussi d'introduire la méthode que

nous proposons dans le prochain chapitre.

Comme nous pouvons remarquer toutes les méthodes considèrent les liens comme une indication de complémentarité entre les pages reliées. Pour l'analyse des liens, toutes les méthodes mentionnées considèrent l'existence d'un seul lien ou d'un chemin entre deux pages comme indicateur de complémentarité entre les pages. La pertinence de cet indicateur est en quelque sorte vérifiée/renforcée par l'utilisation d'un deuxième indicateur. Par exemple, l'hypothèse faite par la méthode II peut être lue de la manière suivante : alors qu'un lien entre deux pages est un fort indicateur de complémentarité entre elles, le fait que les URLs de ces deux pages partagent une certaine relation renforce le premier indicateur. Pour ce qui concerne les autres méthodes, le deuxième indicateur, nous le noterons *indicateur de renforcement*, est basé sur la similarité entre les contenus des pages et sur la présence de termes des requêtes dans les pages.

Nous considérons l'indicateur de renforcement comme un indicateur d'appui au premier, l'indicateur basé sur la structure des liens, qui lui est fondamental dans l'analyse de la complémentarité. Dans toutes les méthodes évoquées, le premier facteur concerne l'existence d'un simple lien ou d'une séquence de liens entre deux pages. Cependant, comme nous l'avons déjà souligné, les liens dans le Web sont très hétérogènes. Par conséquent nous croyons qu'un simple lien ou une séquence de liens est un indicateur très faible de complémentarité — tout en en étant un. Nous pensons que d'autres *motifs* dans la structure du graphe sous-jacent au site analysé pourraient être exploités et s'intégrer dans le premier indicateur basé sur l'analyse des liens.

La méthode que nous décrivons dans le prochain chapitre intègre une analyse plus riche de la structure dans le voisinage des deux pages afin d'estimer une éventuelle complémentarité entre elles. En plus des chemins entre deux pages nous utilisons aussi des techniques issues de la bibliométrie qui utilisent le couplage bibliographique et les co-citations entre deux documents. Cette utilisation plus approfondie de la structure constitue l'une des principales différences entre notre approche et celles que nous avons passées en revue dans la section 2.2.3.

2.3 Synthèse

Dans ce chapitre nous avons passé en revue différentes applications de l'analyse de liens avant d'introduire l'application qui nous intéresse dans ce rapport, la découverte d'unités logiques d'information. Nous avons introduit notre problématique mais aussi analysé et comparé certaines études concernées par elle. Nous avons dégagé un ensemble d'aspects afin de pouvoir mieux comparer les approches proposées. Cependant, aucune des approches n'a été réellement évaluée dans le contexte d'un moteur de recherche. Ainsi, nous n'avons guère d'indications claires sur les bonnes et mauvaises caractéristiques des approches lors de leur intégration dans un moteur de recherche.

Après l'étude de ces méthodes nous avons pu nous forger une opinion sur les caractéristiques que nous souhaiterions retrouver dans une méthode de découverte de pages complémentaires. Par exemple, malgré le fait que quatre parmi les six méthodes citées utilisent la structure de répertoire codée dans l'URL des pages, nous pensons que ce choix n'est pas judicieux puisqu'il suppose une organisation hiérarchique des fichiers. Ainsi, dans notre méthode nous n'avons

pas voulu utiliser ce type d'indicateur de complémentarité. Un autre point qui nous a semblé important après cet état de l'art est le type d'analyse qui est faite sur la structure du graphe sous-jacent à l'espace de recherche considéré. Tel que nous l'avons dit plus haut, l'analyse se restreint à l'identification d'un lien ou d'un chemin reliant deux pages. Nous pensons qu'une analyse plus profonde de cette structure peut être faite, même si elle risque de rendre le coût de calcul plus élevé.

Dans le prochain chapitre nous décrivons la méthode que nous proposons pour la découverte d'unités logiques d'information. Certaines caractéristiques de cette méthode figurent dans le tableau 2.2.4. Nous abordons aussi la manière dont l'information découverte concernant la complémentarité des pages est intégrée à un moteur de recherches.

Chapitre 3

Modèle de SRI pour un système hypertexte

Dans ce chapitre nous décrivons nos études concernant le développement d'un système de recherche d'information adapté à un système hypertexte. La grande différence de notre système par rapport à un système traditionnel est la prise en compte de l'éventuel fractionnement d'un document en plusieurs nœuds lorsque ce document est rendu accessible par un système hypertexte[1].

Comme nous l'avons vu dans la section 2.2.3 du chapitre précédent, il existe un certain nombre de travaux dans la littérature qui tiennent compte du fractionnement des documents et ne considèrent pas une page HTML comme étant un document à part entière. Ces travaux considèrent que le contenu d'une page n'exprime pas complètement l'information qu'elle véhicule. Autrement dit, par la seule analyse du contenu d'une page, un acteur (humain ou informatique) n'est pas capable d'appréhender l'information que la page rend accessible. Dans une page il manque souvent de l'information contextuelle. L'information contextuelle d'une page spécifie le contexte de son contenu et est fondamentale pour la compréhension de l'information véhiculée par la page. Les travaux cités plus haut cherchent à découvrir cette information contextuelle.

Nous pouvons considérer le regroupement du contenu d'une page avec son information contextuelle comme étant un document. Or, si l'information contextuelle d'une page ne se trouve pas dans son propre contenu, elle se trouve dans le contenu d'autres pages. Supposons que l'information contextuelle de la page p_i se trouve dans la page p_j. Il est bien probable que seule une petite partie du contenu de p_j exprime l'information contextuelle nécessaire pour comprendre l'information véhiculée par p_i. Or, dans un système hypertexte, un nœud peut faire partie de plusieurs documents différents. Le contenu complet d'une page n'est pas forcé-

[1] Un document peut être fragmenté aussi bien lors de sa création — il aura été créé déjà pour un média hypertextuel — qu'a posteriori s'il s'agit d'adapter un document au média hypertextuel bien qu'étant originellement conçu pour un média traditionnel. Par exemple, le livre bien connu de Rijsbergen [vR79] à été adapté au média hypertextuel à travers le système TACHIR (`http://www.dcs.gla.ac.uk/~iain/keith/data/tachir/`). La version hypertextuelle est disponible sur `http://www.dcs.gla.ac.uk/~iain/keith/index.htm`. Finalement, certains textes peuvent être dès leur création conçus dans les formats traditionnel et hypertextuel. L'ouvrage récent de Rick Belew [Bel00] en est un exemple.

ment en relation avec tous les documents dont elle fait partie. Différentes parties de la page peuvent concerner différents documents. Nous pensons qu'une méthode susceptible d'identifier avec une certaine justesse les parties d'une page liées aux différents documents dont elle ferait partie ferait probablement appel à la technologie du traitement de langage naturel. Or, il est connu que les informations disponibles dans le Web sont d'une manière globale rédigées dans des styles multiples très différents de celui utilisée dans la rédaction d'un document sur support papier[2]. Les règles syntaxiques de la langue ne sont pas respectées. C'est comme si la grammaire d'une langue donnée était différente de la traditionnelle ou multiple dans les documents écrits dans cette langue et publiés sur le Web. Souvent dans le texte on ne retrouve plus les éléments et les règles d'une grammaire traditionnelle : par exemple on ne retrouve plus systématiquement des phrases, des verbes demandant un complément d'objet sont dépourvus d'objet, des verbes personnels sont utilisés sans sujet, etc. [Ami97][Ami00]. Ainsi, la technologie actuelle développée dans le domaine du traitement du langage naturel pourrait s'avérer peu utile dans l'analyse du texte contenu dans les pages Web. Elle est basée sur un discours qui respecte un type de grammaire différente de celle du Web.

Cela constitue l'une des raisons pour lesquelles la découverte de documents dans un hypertexte s'avère si complexe. Ainsi, les travaux décrits dans la section 2.2.3 ainsi que celui que nous décrivons dans ce chapitre ne cherchent pas exactement à identifier des *documents* dans le Web. Ces travaux visent plutôt à identifier des pages contenant de l'information *contextualisant* — par des parties — le contenu d'autres pages. Ainsi, pour une page donnée l'objectif consiste à trouver les autres pages qui la contextualiseraient. L'ensemble composé de la page centrale et de celles qui la contextualisent correspond à ce que nous avons appelé *unité logique d'information*, ULI, dans le chapitre précédent. Dans ce chapitre, nous appelons les pages qui fournissent le contexte à une page donnée, disons p_i, **les pages complémentaires** de p_i.

Dans la suite nous proposons deux méthodes pour la découverte des pages complémentaires à des pages composant des sous-ensembles du Web. Plus bas nous précisons ce que nous entendons par (i) le Web en termes d'information et (ii) un sous-ensemble du Web. La deuxième méthode est en réalité une extension de la première. Nous avons choisi de démontrer l'utilité de la découverte des pages complémentaires dans le fonctionnement des moteurs de recherche. Ainsi, nous réservons deux sections dans ce chapitre pour détailler l'intégration de l'information concernant la complémentarité des pages dans le fonctionnement du système de recherche d'information sous-jacent aux moteurs de recherche.

Le chapitre s'organise de la manière suivante : dans la première partie nous définissons le type de données concerné par les méthodes de découverte des pages complémentaires — ce que nous appelons un sous-ensemble du Web ; dans la deuxième partie nous présentons la première méthode que nous proposons pour la découverte des pages complémentaires et nous détaillons l'intégration de l'information découverte dans le fonctionnement d'un système de recherche d'information qui se servira de cette information. Dans la troisième partie, nous détaillons la deuxième méthode que nous proposons pour la découverte des pages complémentaires et l'intégration, différente de la première, de cette information dans un moteur de recherche. Dans la dernière partie nous évoquons des questions liées aux limitations des

[2]On peut toujours imaginer une sous partie du Web, e.g. un ensemble de sites, qui respecterait un style de rédaction donné. Mais cela reste marginal par rapport à l'immensité du Web.

méthodes présentées, à la complexité des algorithmes associés aux méthodes, à la capacité d'adaptation des méthodes à des quantités d'information bien supérieures à celles pour lesquelles les méthodes ont étés originellement conçues et, finalement, nous évoquons d'autres applications pouvant bénéficier de l'information contextuelle des pages.

3.1 Définition du corpus de données

Bien entendu, l'information hypertextuelle à laquelle nous nous intéressons ici est celle disponible dans le Web. Dans cette étude nous considérons le Web comme étant un ensemble de ressources distribuées et faiblement couplées ; ces ressources pouvant être considérées comme des systèmes hypertextes locaux. Ainsi, on peut décomposer le Web en un ensemble de systèmes hypertextes plus petits, chacun d'eux étant rendu accessible à travers un *serveur HTTP*, communément appelé un *serveur web*. Un serveur web constitue une application capable d'interagir avec des applications clientes à travers l'utilisation du protocole HTTP qui permet basiquement le transfert de fichiers. Le protocole HTTP comporte un nombre réduit de primitives ; la plus importante d'entre elles étant la primitive `GET`. Cette primitive est utilisée par les clients pour demander des fichiers aux serveurs. N'importe quel type d'application reconnaissant le protocole HTTP peut constituer un client web. Comme exemple de client nous pouvons citer les navigateurs (e.g. *Netscape*, *Internet Explorer*, *Lynx*), les aspirateurs (e.g. *wget*, *pavuk*[3]), les robots utilisés par les moteurs de recherche (e.g. *Scooter*, *Googlebot*[4]), etc.

Les fichiers stockés dans les serveurs Web peuvent avoir, a priori, n'importe quel format. On peut y trouver des fichiers correspondant à des textes, des fichiers correspondant à des images, à des sons, à des vidéos, des fichiers binaires correspondant à des programmes, des fichiers compressés, des fichiers archives, etc. Cependant, dans nos travaux nous nous sommes intéressés aux fichiers correspondant à des textes. Dans les tests que nous décrivons dans le chapitre 4, ces fichiers ont un *mime-type*[5] `text/plain` [FB96] ou `text/html` [PH97]. Ces derniers représentent d'une manière globale ce que l'on appelle couramment *page web*[6].

Les pages web sont reliées grâce à un élément particulier du langage HTML, la balise `<A>`, dont l'utilité principale est d'insérer dans une page un lien vers une autre page. Les mobiles pour placer les liens dans une page HTML sont aussi divers que la sémantique même des liens. Lorsqu'un lien d'une page est activé par l'utilisateur, le navigateur affiche la page cible de ce lien[7]. Il faut remarquer que nous ne considérons pas le fameux Web invisible, les pages

[3] Les aspirateurs *wget* et *pavuk* sont disponibles sur `www.gnu.org/directory/wget.html` et `www.idata.sk/~ondrej/pavuk/` respectivement.

[4] *Scooter* et *Googlebot* sont les robots utilisés par les moteurs *Altavista* et *Google* respectivement.

[5] *Multipurpose Internet Mail Extensions types.*

[6] Les pages web telles que l'on les voit rendues par un navigateur traditionnel peuvent être composées par plusieurs fichiers et non pas par un seul fichier lorsque ce fichier est du type text/html. Un fichier de ce type peut inclure ou pointer sur d'autres fichiers de types autre que plain/html qui sont aussi utilisés par le navigateur afin d'afficher une page. L'élément `<IMG>` du langage HTML par exemple permet qu'une page associée à un fichier de type html/plain inclut une image stockée dans un autre fichier.

[7] La cible d'un lien n'est pas toujours une page. La cible peut être n'importe quel type de fichier, image, son, texte, etc. Lorsque l'on clique sur un lien dont la cible n'est pas une page le navigateur peut agir de différentes façons selon sa configuration. Par exemple, le navigateur peut être configuré pour lancer une deuxième application qui pourra elle traiter le fichier cible du lien. Dans ce type de configuration, on fait appel

dont l'accès est limité d'une façon ou d'une autre[8]. Certains considèrent les pages générées dynamiquement — par exemple à travers des *cgi-bins* interrogeant une base de données et générant à la volée une page incluant la réponse à la requête — comme faisant partie du web invisible. Nous ne considérons pas non plus les pages dynamiques dans nos études.

Comme nous l'avons dit plus haut nous considérons le Web visible comme étant un hypertexte distribué. Ce sont les liens entre les pages qui fournissent le *caractère hypertextuel* au Web. La représentation la plus traditionnelle du Web consiste à le considérer comme un graphe orienté dont les nœuds représentent les pages et les arcs représentent les liens (hypertextuels) entre les pages. Il s'agit du graphe de citations déjà introduit au début du chapitre 2. La version non-orienté de ce graphe n'est pas nécessairement connexe. Cela revient à dire qu'il peut y avoir un sous-ensemble de pages dans le Web qui ni ne pointent ni ne sont pointées sur/par d'autres pages en dehors du sous-ensemble auquel elles appartiennent[9].

Dans l'étude que nous décrivons dans ce chapitre nous nous intéressons seulement à des parties du graphe représentant le Web et non pas au graphe entier. Il est important de signaler que cela est un vrai choix et non pas une conséquence du grand volume de données disponible dans le Web. Au contraire, les moteurs de recherche traditionnels s'intéressent au graphe complet du Web, cependant, ils n'en traitent dans la pratique qu'une partie. Cela est dû à des contraintes de temps et d'espace auxquelles ces systèmes sont soumis.

Le type de sous-graphe qui nous intéresse ici est celui dont les nœuds sont associés à des pages situées dans un même *site Web*. Il nous faut donc définir ce que nous considérons comme un site.

La définition la plus simplificatrice qui soit pour définir un site est celle décrivant l'ensemble de pages dont les URL partagent le préfixe composé du tuple <*nom de machine hôte*, *numéro de port*>. Ce tuple identifie un serveur HTTP écoutant sur le port *numéro de port* de la machine associée au nom logique *nom de machine hôte*[10]. Ainsi, les pages ayant pour URL `http://www.aaa.bbb:80/a.html` et `http://www.aaa.bbb:80/b.html` appartiendraient à un même site. Cependant, selon cette définition toutes les pages dont les URLs ont comme préfixe `http://www.geocities.com:80/` appartiennent à un même site. Or, d'une manière assez intuitive on constate que sous `http://www.geocities.com:80/` il existe une myriade de sites différents. On peut constater ce même type de situation chez d'autres *hébergeurs* de sites web tels que *tripod* (`http://members.tripod.com:80/`) et *multimania* (`http://membres.lycos.fr:80/`).

---

aux *plug-ins* qui constituent des applications sachant traiter des fichiers de types particuliers.

[8]Nous pouvons classer les pages appartenant au Web invisible sous deux grandes catégories : (i) les pages hébergées par des serveurs installés dans des machines derrière un pare-feu ; (ii) les pages dont l'accès est soumis à un processus d'autorisation, ce processus la plupart du temps se matérialisant par la déclinaison au serveur d'un nom d'utilisateur et d'un mot de passe.

[9]Une telle page peut cependant être accédée si elle est indexée par un moteur de recherche. En effet, en plus des robots, certains moteurs de recherche collectionnent les pages qu'ils indexeront à la suite grâce à un formulaire d'inscription volontaire de pages qui peut être utilisé par les auteurs.

[10]À tout nom logique est associé une adresse IP qui identifie une machine. L'association est faite par un serveur de noms de domaine, DNS (*Domain Name Server*).

Nous croyons que l'affirmation selon laquelle il y a plusieurs sites sous `http://www.geocities.com` vient de l'observation que les pages dont les URLs partagent le préfixe `http://www.geocities.com:80/jmdesbois` (histoire de la ville d'Aix en Provence) et celles dont les URLs partagent le préfixe `http://www.geocities.com/mez_sv/` (voitures exotiques) n'ont aucun rapport entre elles si ce n'est d'être physiquement hébergées sur la même machine. Cette capacité d'identification des frontières d'un site qui existe chez n'importe quel utilisateur du Web semble donc relever d'une espèce de cohérence observée dans les pages qui composent un site. Cette cohérence peut être thématique mais nous croyons qu'elle pourrait se trouver dans une dimension autre que thématique. En considérant la dimension thématique, les pages d'un site — tel qu'un utilisateur moyen du Web le conçoit — traiteraient d'un unique thème ou sinon de thèmes divers qui pourraient être englobés dans un thème unique plus général[11]. Cependant, si deux ensembles d'URLs traitent d'une thématique commune, cela n'indique pas nécessairement que ces deux ensembles forment un seul site. Or, dans le Web on retrouve de nombreux sites qui s'intéressent à une thématique commune ; les répertoires thématiques comme *Dmoz* (`www.dmoz.org`) ou *Yahoo!* (`www.yahoo.com`) étant des services de recherche qui organisent les sites selon leur proximité thématique.

À notre connaissance, un nombre réduit des travaux scientifiques, e.g. [CHR01] [ATH00] [THA99] [LKV00], relèvent la question évoquée dans les deux paragraphes précédents. Dans le contexte de ces travaux un site est considéré comme étant un ensemble de pages associées à une thématique commune, maintenues par une seule personne ou un seul groupe de personnes, et **publié comme un ensemble cohérent d'information**. Cette définition est donc cohérente avec l'intuition commune que derrière une URL comme `http://www.geocities.com/` il n'existe pas un seul site.

Cependant, l'identification des sites formant un ensemble cohérent d'information est bien problématique. Puisque la définition d'un site reste assez floue, les rares algorithmes proposés pour leur identification se basent sur des heuristiques dont l'efficacité est difficilement estimable. [LKV00] propose une heuristique basée sur l'url des pages (les noms de répertoire et de fichier ainsi que la hiérarchie de répertoires induite), le texte de l'ancre des liens présents dans les pages et la structure du graphe sous-jacent aux pages pour identifier les différents sites qui pourraient former les pages partageant le tuple <*nom de machine hôte, numéro de port*>. Dans [THA99] Terveen propose une autre heuristique basée sur l'url des pages avec des règles spécifiques pour l'analyse de sites particuliers tels que `http://members.tripod.com:80/` et `http://www.geocities.com:80/`.

Le point commun des heuristiques proposées est de supposer que deux pages appartenant à un même site sont toujours hébergées par un même serveur HTTP. Afin de savoir si deux pages sont servies par un même serveur les deux heuristiques que nous venons de citer vérifient si les tuples <*nom de machine hôte, numéro de port*> de deux URLs sont identiques.

[11] Par exemple, le site de l'École nationale supérieure des mines de Saint-Étienne accessible sur `www.emse.fr` pourrait être décomposé en au moins quatre sites différents étant donné qu'il existe quatre ensembles de pages, chacun étant associé à l'un des quatre centres de l'école et traitant ainsi de thématiques différentes. Cependant, du fait que ces quatre ensembles de pages forment un ensemble cohérent d'information — il s'agit des pages en rapport avec l'École des mines de Saint-Étienne — ils peuvent être considérés comme des composants d'un seul site.

S'ils le sont les deux pages **peuvent** appartenir à un même site. Dans le cas contraire, les heuristiques considèrent que les deux pages appartiennent nécessairement à deux sites différents. Nous allons voir plus bas que cette vérification n'est pas suffisante pour conclure que deux pages ne sont pas servies par un même serveur HTTP.

D'après l'hypothèse de base commune aux heuristiques, deux pages dont les URLs sont `http://aa.com:80/a.html` et `http://bb.com:80/b.html` ne peuvent pas appartenir à un même site. Or, plusieurs *noms logiques* peuvent être associés à une même adresse IP, l'un des noms étant le nom canonique associé à l'adresse IP donnée et les autres noms étant les *alias* de ce nom canonique. `bb.com` peut être un alias de `aa.com` et ainsi les deux pages pourraient a priori être hébergées par un même serveur ainsi elles pourraient appartenir à un même site. De ce fait, les heuristiques devraient ainsi vérifier les alias des noms de machines hôtes des URLs des pages. Il s'avère qu'il n'est pas toujours possible de connaître exactement tous les alias d'un nom logique donné.

D'autre part, différentes adresses IPs peuvent être associées à une même machine. Ainsi, deux URLs dont les noms des machines hôtes seraient associées à deux adresses IPs différentes, pourraient représenter deux pages appartenant à un même site. Il est en effet impossible de connaître toutes les adresses IPs associées à une machine donnée. Ainsi, on ne peut pas toujours savoir avec certitude si deux pages sont hébergées par un même serveur HTTP ou non.

La méthode que nous développons dans la suite s'intéresse à des sites compatibles avec la définition selon laquelle un site représente un ensemble cohérent d'information. Malgré le manque d'informations concernant l'homogénéité/l'hétérogénéité du type d'information disponible dans les serveurs HTTP actuellement, nous croyons que des cas similaires à ceux représentés par les services disponibles sur `www.geocities.com` ou `perso.free.fr` sont marginaux dans le Web.

À l'instar de Craswell dans les expérimentations décrites dans [CHR01], nous supposons que les sites que nous avons utilisés dans nos expérimentations sont compatibles avec la définition de site que nous avons adoptée. Dans le chapitre 4, nous détaillons la collection de tests utilisée et la manière dont les sites ont été identifiés.

Pour résumer, notre méthode s'intéresse à découvrir l'information contextuelle des pages regroupées dans un sous-graphe du Web visible et non pas à découvrir ce même type d'information pour toutes les pages du Web visible. Les pages composant le type de sous-graphe qui nous intéresse composent un site dans le sens où elles représentent un ensemble cohérent d'information et sont associées à une thématique générale unique.

Dans notre étude nous utilisons la représentation la plus traditionnelle qui soit pour représenter le Web. Comme cité plus haut, cette représentation consiste dans un graphe orienté où les nœuds représentent les pages HTML et les arcs représentent les liens hypertextuels. On peut aussi appeler ce graphe le *graphe de citations*. Dans la figure 3.1 nous illustrons cette représentation ainsi que le type d'information concernée par la méthode de découverte d'information contextuelle que nous décrivons dans la suite.

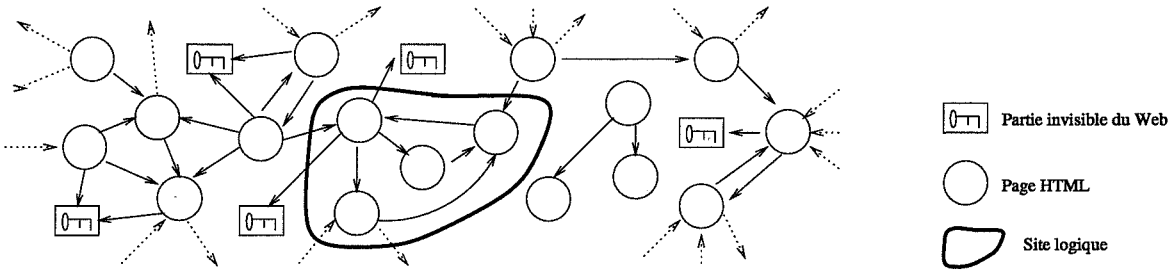

FIG. 3.1: Un site logique dans le Web.

3.2 La découverte des pages complémentaires et des contextes

Rappelons que les pages complémentaires d'une page donnée, disons p_x, sont les pages dont les contenus contextualisent l'information véhiculée par p_x. Tel que nous l'avons remarqué plus haut, dans certains cas, il se peut que ce ne soit pas les contenus entiers des pages complémentaires de p_x qui apportent le contexte de l'information véhiculée par p_x mais seulement certaines de leurs parties. Des heuristiques peuvent être utilisées pour essayer d'identifier ces parties complémentaires ou, alternativement, seules certaines parties — définies a priori — d'une page pourraient être considérées comme étant de l'information contextuelle pour une autre page. Nous y reviendrons dans la section suivante.

Dans ce chapitre nous proposons deux approches pour trouver l'information contextuelle des pages. Dans cette section nous décrivons la première et dans la section 3.4 nous détaillons la seconde qui est en fait une extension de la première.

Ces deux méthodes sont basées sur une fonction que nous avons définie pour estimer dans quelle mesure deux pages se complémentent. Nous pouvons définir la complémentarité entre deux pages comme étant une **relation binaire floue** sur l'ensemble de pages analysées. La relation est **floue** puisqu'une page est complémentaire d'une autre à un certain degré. Nous noterons $Compl(p_x, p_y)$ la mesure de complémentarité de p_x pour p_y, i.e. cette valeur estime combien p_x complémente p_y. Les formules que nous allons proposer dans la suite feront que cette relation de complémentarité sera symétrique, c.-à-d. $Compl(p_x, p_y) = Compl(p_y, p_x)$.

Cette symétrie est simplificatrice. Dans notre contexte, si p_x complémente p_y, p_x contient de l'information qui sert de contexte au contenu de la page p_y. Cela n'indique pas nécessairement que p_y contienne aussi de l'information contextuelle pour le contenu de p_x et, encore moins, à un même degré, même si dans un support hypertextuel nous pouvons imaginer ce type de situation.

La première approche que nous décrivons dans la suite peut être résumée de la manière suivante : d'abord la complémentarité entre toutes les paires de pages analysées est calculée ; ensuite, l'identification d'ensembles contenant des pages ayant des degrés de complémentarité élevés entre elles est effectuée. Chacun de ces ensembles peut être d'ailleurs considéré comme un volume d'information auto-explicatif, ce que nous avons appelé dans le dernier chapitre *unité logique d'information.* Une fois que les ensembles ont été découverts, une fonction est

appliquée à chacun de ces ensembles de sorte à en déduire l'information contextuelle des pages composant l'ensemble. Nous avons appelé cette fonction *la fonction d'abstraction.*

Dans ce qui suit nous décrivons la mesure de complémentarité (section 3.2.1) ; la méthode pour identifier les ensembles de pages complémentaires (section 3.2.2) et finalement la fonction d'abstraction qui génère l'information contextuelle des pages (section 3.2.3).

3.2.1 La mesure de complémentarité

Considérons le graphe de citations $G = (N, A)$ sous-jacent aux pages concernées par l'analyse de complémentarité[12]. La complémentarité entre deux nœuds de G est une fonction de leur *proximité structurelle* et de la similarité du contenu des pages associées aux nœuds. Nous définissons ainsi la complémentarité entre deux nœuds, N_i et N_j de G :

$$Compl(N_i, N_j) = \alpha \cdot Prox_{struc}(N_i, N_j) + (1 - \alpha) \cdot Sim_{cont}(N_i, N_j) \tag{3.1}$$

Dans les expérimentations que nous décrivons dans le prochain chapitre nous avons varié la valeur de α entre 0 et 1 de sorte à avoir des indications sur une éventuelle influence plus élevée de l'un des deux facteurs de la dernière équation dans la découverte de complémentarité entre deux pages. Autrement dit, ce paramètre permet d'analyser si l'un des deux aspects (celui lié au contenu des pages ou celui lié à la structure les reliant) apporte plus d'indices que l'autre dans l'estimation d'une éventuelle complémentarité entre deux pages.

Une question qui peut être soulevée ici est l'éventuelle dépendance de l'importance relative des 2 termes de l'équation 3.1 par rapport à certaines caractéristiques (e.g. la densité du graphe) des hypertextes analysés. Nous essayons de répondre aussi à cette question à travers l'analyse des résultats des expérimentations reportées dans le prochain chapitre.

3.2.1.1 Similarité du contenu

Le contenu d'un nœud[13] est représenté à l'aide du modèle vectoriel introduit par Salton au début des années 80 [SM83]. Dans ce modèle, le contenu d'un document appartenant à un ensemble de documents est représenté par un vecteur dans un espace de ρ dimensions où ρ est le nombre de termes différents présents dans les n documents qui forment la collection de données analysée.

Notons qu'ici la sémantique du terme *document* n'est pas la même que celle que nous avons définie dans l'introduction de ce manuscrit (section 1.2.1). Un document dans ce contexte-ci est un volume d'information dont les frontières sont connus, e.g. la page d'un livre, un livre, une notice bibliographique, un article de journal, une page HTML, etc. Son volume d'information n'est pas forcément auto-explicatif d'où la différence avec la définition donnée plus

[12]Rappelons que dans le graphe de citation sous-jacent à l'ensemble des pages considérées, il existe au plus un arc reliant un nœud, disons N_i, à un autre nœud, disons N_j. Par conséquent, plusieurs hyperliens menant de la page représentée par N_i à la page représentée par N_j sont représentés dans le graphe de citation par un seul arc menant de N_i à N_j.

[13]Dorénavant, nous allons faire référence au *contenu de l'entité représentée par un nœud*, par tout simplement, *le contenu du nœud.* Dans cette section l'entité associée à un nœud est une page. Dans la section 3.4 les entités associées aux nœuds seront des contextes et non plus des pages.

haut.

Dans notre contexte, les documents représentés à l'aide du modèle vectoriel de Salton sont les pages HTML appartenant à un même site. Chacune de ces pages est représentée par un vecteur dans un espace de ρ dimensions où ρ est le nombre de termes différents présents dans les pages appartenant au site concerné par l'analyse de complémentarité. Considérons dans la suite $T = \{t_1, \cdots, t_\rho\}$ l'ensemble des termes différents dans la collection des pages considérées.

Les coordonnées du vecteur représentant un document sont les poids associées aux différentes termes de l'espace T. L'idée est que le poids associé à un terme d'un document représente dans quelle mesure ce terme est représentatif du contenu sémantique du document. La similarité entre les contenus de deux documents, $Sim_{cont}(N_i, N_j)$, est calculée à l'aide de la mesure du cosinus de l'angle formé entre les vecteurs représentant les deux documents. Ainsi la similarité est maximale — égale à 1 — lorsque l'angle formé par $\overrightarrow{N_i}$ et $\overrightarrow{N_j}$ est de 0 degré. La similarité est minimale — égale à 0 — lorsque les deux vecteurs forment un angle de 90 degrés, i.e. lorsque les vecteurs sont orthogonaux. La formule qui permet de calculer $Sim_{cont}(N_i, N_j)$ est donc la suivante :

$$Sim_{cont}(N_i, N_j) = \frac{\sum_{k=1}^{\rho} w_{ik} \cdot w_{jk}}{\sqrt{\sum_{k=1}^{\rho} w_{ik}^2 \cdot \sum_{k=1}^{\rho} w_{jk}^2}} \tag{3.2}$$

où w_{ik} est le poids associé au terme t_k dans la représentation du document N_i. En ce qui concerne le calcul du poids d'un terme pour un document donné, d'innombrables approches existent. La presque totalité de ces approches se basent sur la fréquence d'occurrence des termes à l'intérieur des documents de la collection considérée.

Au cours des dernières années, plusieurs formules de pondération ont été introduites et testés dans le contexte du modèle probabiliste de recherche d'information OKAPI. La formule de pondération que nous avons utilisée est l'une d'entre elles [RWHB+94]. Elle a réussi une bonne performance dans les expérimentations menées lors des conférences TREC de ces dernières années [RWB00] et fut ainsi réutilisée par d'autres systèmes de recherche d'information. Il s'agit d'une pondération basée sur la fréquence d'apparition des termes à l'intérieur des documents. Sa particularité se trouve dans son facteur de normalisation dont le but est de compenser les grandes fréquences de termes observées dans les documents longs. Nous savons que si les poids ne sont pas normalisés, les documents longs sont favorisés lors de l'estimation de leur degré de pertinence à une requête, au détriment des documents courts (où les fréquences d'occurrence des termes ne sont pas élevées). La formule de pondération utilisée est la suivante :

$$w_{is} = (k_1 + 1) \cdot \frac{tf_{is}}{K + tf_{is}} \quad \text{où} \quad K = k_1 \cdot \left((1 - b) + b \cdot \frac{l_i}{avdl} \right) \tag{3.3}$$

Dans cette équation tf_{is} représente la fréquence d'occurrence du terme t_s dans le contenu du document N_i ; l_i représente la longueur du document N_i (somme des fréquences d'occurrence de tous les termes du document N_i) et *avdl* représente la longueur moyenne des documents.

Finalement, b et k_1 sont des constantes. Dans les tests décrits dans le prochain chapitre b et k_1 ont pour valeur 0.7625 et 1.5 respectivement.

L'hypothèse sous-jacente au second terme de l'équation 3.1, celui lié au contenu des documents, que nous venons de définir à l'aide des équations 3.2 et 3.3 est la suivante : lorsque le contenu de deux documents sont proches thématiquement, la probabilité que leurs contenus soient complémentaires mutuellement est plus élevée que lorsque leurs contenus diffèrent. Autrement dit, nous supposons que plus la similarité entre deux documents est grande, plus élevée est la probabilité que leurs contenus soient complémentaires.

Des hypothèses similaires ont été utilisées dans d'autres travaux. Par exemple, Hearst dans [HP93] propose une méthode de division automatique d'un document textuel en passages — chaque passage traitant d'une seule thématique — basée sur la similarité des contenus de fenêtres de textes consécutives dans le document. L'hypothèse étant que deux passages consécutifs doivent être complémentaires à un certain degré. Cette complémentarité est identifiée à travers la similarité textuelle des deux passages. Comme évoqué dans la section 2.2.3, d'autres travaux [TMKT98] [KK99] [HSDT99] dans la thématique de la découverte d'unités logiques d'information adoptent aussi des hypothèses semblables.

3.2.1.2 Proximité structurelle

Le premier terme de l'équation 3.1 est lié à la proximité structurelle entre deux nœuds, N_i et N_j, du graphe sous-jacent à l'espace d'information sur lequel l'analyse de complémentarité est faite. Nous considérons la proximité structurelle entre deux nœuds du graphe, $Prox_{struc}(N_i, N_j)$, comme étant une fonction (i) du nombre d'ancêtres communs aux deux nœuds (S_{ij}^{anc}) (équation 3.4), (ii) du nombre de descendants communs aux deux nœuds (S_{ij}^{des}) (équation 3.5) et (iii) de l'inverse de la longueur des plus courts chemins connectant les deux nœuds (S_{ij}^{spl}) (équation 3.6).

Pour ce qui concerne les ancêtres et les descendants d'un nœud, nous supposons que plus deux nœuds ont d'ancêtres et de descendants en commun, plus grande est la probabilité que leurs contenus soient complémentaires. Dans un graphe orienté, un nœud N_i est un ancêtre (resp. *descendant*) d'un autre nœud N_j s'il existe dans ce graphe un chemin menant de N_i à N_j (resp. N_j *à* N_i). Ces hypothèses basées sur le nombre d'ancêtres et de descendants en commun de deux nœuds sont inspirées des techniques connues utilisées dans le domaine de la bibliométrie. Les méthodes du couplage bibliographique [Kes63] [Wei74] et des co-citations [Mar73] [Sma73] ont comme objectif d'identifier des documents thématiquement proches à travers l'analyse de leurs citations. Les hypothèses considérées par les deux méthodes sont que deux documents sont thématiquement proches si, respectivement : (i) ils partagent un grand nombre de citations et (ii) ils sont souvent co-cités par d'autres documents. Alors que dans ces méthodes seul le voisinage immédiat d'un document, i.e. les documents qui citent ou sont cités par un document donné, est analysé, nous considérons un voisinage plus étendu par l'utilisation des ancêtres et des descendants. Une autre différence fondamentale entre l'analyse de citations dans la bibliométrie et l'analyse de liens dans le Web est que dans le premier cas les liens ont une sémantique relativement bien définie, alors que dans le second les liens peuvent avoir des sémantiques bien différentes et diverses [Bel00, chap. 6]. Finalement,

vu que les pages web ne sont pas figées mais sont en constante évolution, deux pages peuvent se citer mutuellement ce qui est impossible avec les documents étudiés dans la bibliométrie.

En revenant à l'équation 3.1, les facteurs S_{ij}^{anc} et S_{ij}^{des} que nous utilisons pour estimer la complémentarité structurelle entre deux nœuds sont calculés d'après les formules suivantes :

$$S_{ij}^{anc} = \sum_{x \in Anc} \frac{1}{2^{(spl_{xi}^{\bar{j}} + spl_{xj}^{\bar{i}})}} \tag{3.4}$$

$$S_{ij}^{des} = \sum_{x \in Desc} \frac{1}{2^{(spl_{ix}^{\bar{j}} + spl_{jx}^{\bar{i}})}} \tag{3.5}$$

où *Anc* et *Desc* désignent respectivement les ensembles des ancêtres et descendants communs aux nœuds N_i et N_j et $spl_{ij}^{\bar{k}}$ correspond à longueur du plus court chemin menant du nœud N_i au nœud N_j sans passer par le nœud N_k. Il est important de faire deux remarques à propos de ces deux dernières équations. La première consiste à souligner que le degré de parenté d'un ancêtre ou descendant en commun est prise en compte dans le calcul. Nous considérons qu'un ancêtre ou descendant direct commun à deux nœuds est un indicateur de complémentarité plus forte qu'un ancêtre ou descendant de deuxième ou troisième *génération*. La deuxième remarque consiste à noter que nous tenons compte du cas où l'un des deux nœuds, disons N_i, est ancêtre de l'autre, i.e. N_j. Dans ce cas, tous les ancêtres de N_i sont aussi ancêtres des N_j. Il en résulterait un degré de complémentarité très élevé entre tous les couples de nœuds liés par un chemin dans le graphe analysé. Les facteurs de type $spl_{xj}^{\bar{i}}$ évitent cette dernière situation puisque si le nœud N_x est par exemple un ancêtre commun à N_i et N_j, $spl_{xj}^{\bar{i}}$ représente le plus court chemin entre le nœud N_x et le noeud N_j **sans passer** par N_i. Ainsi, si N_i est ancêtre de N_j et que le seul chemin existant entre l'ancêtre en commun N_x et N_j passe par N_i, $spl_{xj}^{\bar{i}}$ vaudra la valeur infinie et, par conséquent, l'ancêtre en commun N_x n'aura pas contribué à augmenter la valeur S_{ij}^{anc} de l'équation 3.4.

Finalement, le dernier facteur, S_{ij}^{spl}, porte sur la longueur des chemins les plus courts entre les nœuds. Nous supposons que s'il existe un chemin connectant deux nœuds, plus ce chemin est court plus grande est la probabilité que les deux nœuds soient complémentaires. Ainsi, nous définissons S_{ij}^{spl} comme une fonction des longueurs des deux plus court chemins connectant les nœuds N_i et N_j (un chemin menant de N_i à N_j et un autre menant de N_j à N_i). Si l'un des deux chemins n'existe pas la longueur correspondante vaudra la valeur infinie. La valeur S_{ij}^{spl} est calculée par :

$$S_{ij}^{spl} = \frac{1}{2^{spl_{ij}}} + \frac{1}{2^{spl_{ji}}} \tag{3.6}$$

où spl_{ij} correspond à la longueur du plus court chemin menant du nœud N_i au nœud N_j. Notons que d'après l'équation 3.6, S_{ij}^{spl} et S_{ij}^{spl} ont la même valeur.

Finalement les valeurs des trois facteurs décrits ci-dessus, S_{ij}^{anc}, S_{ij}^{desc} et S_{ij}^{spl}, sont normalisées entre $[0, 1]$ puis la proximité structurelle entre deux nœuds, $Prox_{struc}(N_i, N_j)$, est calculée par une combinaison linéaire des trois facteurs :

$$Prox_{struc}(N_i, N_j) = \beta \cdot S_{ij}^{anc} + \gamma \cdot S_{ij}^{des} + \lambda \cdot S_{ij}^{spl} \quad (3.7)$$

Les poids β, γ et λ permettent de varier l'influence de chaque facteur dans le calcul de la proximité structurelle entre deux nœuds. Nous les avons introduits afin d'essayer d'observer s'il existe parmi ces facteurs structuraux certains qui s'avéreraient plus révélateurs que d'autres vis-à-vis de la complémentarité entre deux nœuds. À l'instar du poids α de l'équation 3.1, nous pouvons aussi soulever la question de savoir si les valeurs optimales de ces poids sont ou non très dépendantes des caractéristiques de l'ensemble de pages considéré dans l'analyse de complémentarité. Cela est l'une des questions auxquelles on essayera de répondre avec les résultats des expérimentations décrites dans le prochain chapitre.

3.2.2 Le regroupement de pages complémentaires

Dans cette section nous décrivons une manière de grouper les pages complémentaires dans des ensembles selon leurs caractéristiques. Nous proposons une méthode basée sur des techniques de clusterisation. Selon le domaine de recherche concerné, les méthodes de **clusterisation** peuvent être appelées autrement, par exemple **classification automatique**. Ainsi, dans le prochain paragraphe nous essayons de bien définir ce que nous entendons par une méthode de clusterisation.

Dans le domaine très large de la recherche d'information, il existe un sous-domaine appelé la **catégorisation** des documents. Catégoriser un document consiste à lui associer une catégorie. La catégorisation peut être supervisée ou non. Lorsque la catégorisation est supervisée, un ensemble de catégories est défini a priori et le processus de catégorisation consiste à associer les documents à ces catégories prédéfinies. La **catégorisation supervisée** est couramment appelée **classification**. Au contraire, dans la catégorisation non-supervisée, il n'y a pas de catégories prédéfinies. Les documents sont groupés d'après les valeurs d'un ensemble de propriétés choisies pour caractériser les documents. La **catégorisation non-supervisée** est appelée **clusterisation**. Ainsi, dans une catégorisation non-supervisée ou clusterisation, on regroupe dans des clusters des documents appartenant à une même catégorie selon les valeurs de propriétés qui les caractérisent. Une telle catégorie est dépourvue d'un label qui l'identifie ou la résume. Afin d'associer un label à une catégorie on peut faire appel à d'autres méthodes capables de le générer à partir des valeurs des propriétés des documents regroupés dans la catégorie.

Pour éviter la confusion, nous allons dans la suite utiliser le terme clusterisation au lieu de classification automatique. Une bonne introduction aux techniques de clusterisation est proposé dans [And73].

Nous avons implémenté plusieurs algorithmes de clusterisation, tous basés sur une mesure d'association entre les éléments que l'on compte grouper. La mesure d'association utilisée par les deux algorithmes est la mesure de complémentarité définie dans la section précédente. Ainsi, le point de départ des algorithmes que nous avons utilisés est une matrice carrée symétrique M d'ordre n, où n est le nombre de nœuds composant l'espace considéré dans l'analyse de complémentarité, i.e. le nombre de pages HTML considérées. L'élément de la matrice $M(i,j)$ contient la valeur estimée de la complémentarité entre les nœuds N_i et

N_j, i.e. $Compl(N_i, N_j)$. Étant donné que $Compl(N_i, N_j) = Compl(N_j, N_i)$ (cf. équation 3.1) la matrice M est symétrique. Dans la suite nous décrivons deux des algorithmes implémentés.

L'un des algorithmes de clusterisation que nous avons implémenté est celui appelé l'algorithme de l'étoile[14]. L'algorithme exploite la structure du graphe déduit des complémentarités entre les pages calculées selon l'équation 3.1 et non pas celle du graphe original de pages. Le graphe utilisé par l'algorithme est déduit de la matrice M', déduite elle-même de la matrice M originale et d'un seuil que nous avons appelé $seuil_{MM'}$. La définition de la matrice M' est effectuée de la façon suivante : $M'(i,j)$ est positionné à 1 si et seulement si $M(i,j) \geq seuil_{MM'}$ et à 0 sinon. De ce fait, la matrice est aussi une matrice carré symétrique d'ordre n. Cette matrice induit le graphe G' non-orienté sur lequel les clusters seront identifiés. Il existe une arête entre deux nœuds N_i et N_j dans G' si et seulement si $M'(i,j) = 1$. Dans l'implémentation utilisée pour mener les expérimentations décrites dans le prochain chapitre, nous avons choisi pour $seuil_{MM'}$ la valeur moyenne des complémentarités représentées dans la matrice M.

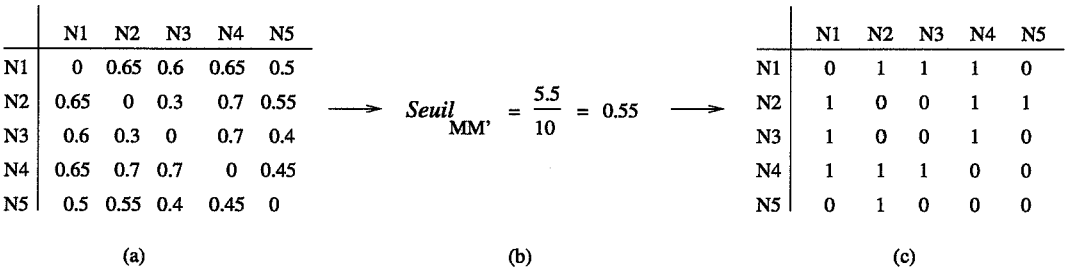

FIG. 3.2: Création de la matrice M' : (a) matrice M originale des complémentarités entre les pages, (b) calcul du seuil à partir de la matrice M, (c) matrice M' dérivée de M à travers l'application du seuil $seuil_{MM'}$.

Les clusters sont définis de la manière suivante. Pour chaque nœud on considère le sous-ensemble composé du nœud lui-même et de tous les nœuds qui lui sont reliés par une arête. Parmi ces sous-ensembles, certains peuvent être inclus dans d'autres, le cas échéant les premiers sont éliminés (figure 3.3). Les sous-ensembles restants sont considérés comme des regroupements de pages complémentaires. Les pages groupées dans un cluster sont donc censées former un volume d'information auto-explicative.

Remarquons d'ailleurs que ce premier algorithme crée des clusters pouvant être non-disjoints. Cette caractéristique est compatible avec l'hypothèse selon laquelle une page peut appartenir à plusieurs documents à la fois.

Un deuxième algorithme de clusterisation que nous avons implémenté afin de regrouper des pages complémentaires est l'algorithme du *lien-complet*[15]. Il s'agit d'un algorithme de

[14] Connu en anglais sous le nom *the star method* [Kow97, chap. 6].
[15] Plus connu sous le nom en anglais : *complete-link*.

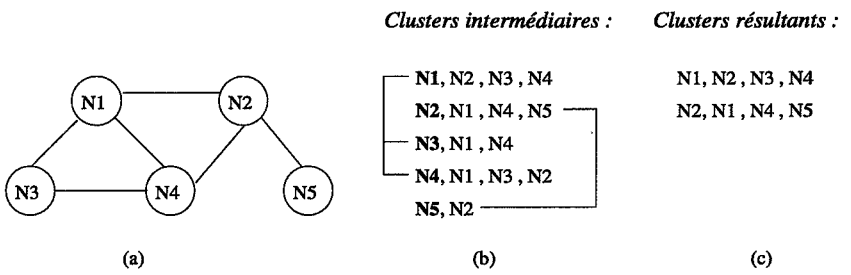

FIG. 3.3: Création des clusters par l'algorithme de l'étoile : (a) graphe déduit de la matrice M', (b) les clusters intermédiaires et leurs regroupements, (c) les clusters résultants.

clusterisation hiérarchique dans la mesure où il génère une hiérarchie de clusters. Chaque niveau de la hiérarchie est associé à un regroupement de deux clusters à une certaine valeur de similarité. Les trois autres algorithmes hiérarchiques les plus connus sont : simple-lien, lien-moyen et l'algorithme de Ward, nous les avons aussi implémentés (cf. annexes).

Tous les algorithmes hiérarchiques peuvent être décrits par un algorithme général :

1. associer un cluster singleton pour chaque élément de l'ensemble à clusteriser ;
2. identifier les deux clusters les plus proches d'après une mesure d'association prédéfinie et les combiner dans un nouveau cluster ;
3. s'il reste plus d'un cluster retourner à l'étape 2.

Les différents algorithmes hiérarchiques se distinguent par la manière dont la similarité entre deux clusters est calculée dans l'étape 2 de l'algorithme décrit ci-dessus. Dans l'algorithme *lien-complet*, la similarité entre deux clusters correspond à la similarité minimale entre toutes les paires d'élements composés d'un élément de chaque cluster. Autrement dit, la similarité entre deux clusters C_x et C_y disjoints est calculée par :

$$Sim(C_x, C_y) = \min_{N_i \in C_x \ , \ N_j \in C_y} Sim(N_i, N_j) \tag{3.8}$$

Dans notre cas la similarité entre deux nœuds N_i et N_j correspond à leur complémentarité disponible dans la matrice de complémentarité M. Reprenons la matrice M utilisée dans l'exemple précédent (figure 3.2(a)). La hiérarchie générée par l'algorithme lien-complet est illustrée dans la figure 3.4.

Les algorithmes hiérarchiques génèrent des clusters disjoints. Ainsi, nous pouvons considérer le résultat d'un tel algorithme comme **n − 1 partitions** de l'ensemble de départ. À chaque fois que deux clusters sont regroupés dans un nouveau cluster, la hiérarchie gagne un nœud supplémentaire ce qui induit une nouvelle partition pour l'ensemble d'éléments de départ. Puisque l'algorithme effectue $n - 1$ regroupements entre les n éléments de départ, l'algorithme génère $n - 1$ partitions. Finalement, à chaque partition est associée une valeur de similarité. Cette valeur correspond à la valeur de similarité sur laquelle le regroupement des deux clusters à l'origine de la nouvelle partition en question a été effectuée. Dans la figure 3.4

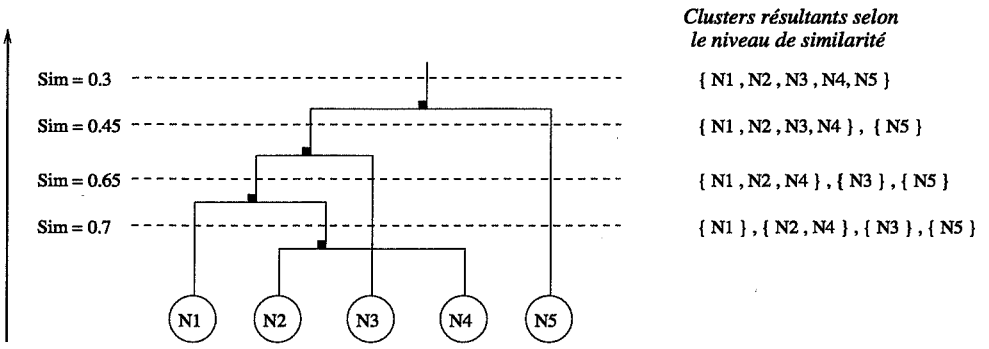

FIG. 3.4: Création des clusters par l'algorithme lien-complet.

nous remarquons quatre niveaux représentant quatre partitions différentes. Ainsi, lorsque l'on utilise un algorithme de clusterisation hiérarchique, on doit disposer d'une règle pour déterminer quelle partition de la hiérarchie sera choisie comme résultat de la clusterisation. Nous pouvons par exemple imaginer des règles basées sur le nombre de clusters dans la partition ou sur le niveau de similarité associé aux partitions. Dans nos expérimentations, nous avons opté pour une règle du premier type : si l'ensemble de départ est composé de n nœuds, nous récupérons comme résultat de la clusterisation la partition composée de $n/2$ cluster(s).

L'algorithme *lien-complet* nous a semblé convenable vu sa caractéristique de générer des petits clusters dont les éléments sont fortement couplés. Cet algorithme est en fait appelé lien-complet puisqu'il garantit que les éléments dans un cluster sont tous liés les uns aux autres avec une similarité minimale. À l'autre extrême, l'algorithme *lien-simple* a tendance à générer des clusters très longs de type ellipsoïde dont les éléments sont très peu liés entre eux. Étant donné que notre but est de découvrir des ensembles de pages qui se complémentent les unes aux autres, la caractéristique de forte liaison entre les éléments est souhaitable.

Plusieurs implémentations de l'algorithme *lien-complet* sont proposées dans la littérature [Def77] [Voo86]. Les différentes implémentations visent bien entendu la réduction de l'espace et du temps nécessaire pour l'exécution de l'algorithme. Cela afin que les algorithmes se comportent bien dans des situations où il faudrait clusteriser des ensembles d'éléments de plus en plus grands. Étant donné que dans notre étude il n'est question de clusteriser que des ensembles de cardinalité pas très élevée — les pages appartenant à un site — la complexité des algorithmes de clusterisation n'a pas vraiment constitué un souci pour nous. Nous avons donc implémenté l'algorithme d'une manière assez simple qui requiert $O(n^2)$ aussi bien en termes d'espace qu'en termes de temps.

En bref, dans les tests nous avons choisi d'utiliser différentes méthodes de clusterisations. Parmi ces méthodes certaines génèrent des clusters disjoints (e.g. la méthode du lien-complet), et d'autres des clusters non-disjoints (e.g. la méthode de l'étoile). L'idée derrière l'utilisation de ces deux types différents de méthodes est d'avoir des indications sur la validité de l'hypothèse selon laquelle une page HTML peut appartenir à plusieurs unités logiques d'information dans le Web.

3.2.3 La fonction d'abstraction

Dans la section précédente nous avons proposé deux manières de regrouper des pages complémentaires dans des ensembles. Deux pages appartenant à un même ensemble sont censées être complémentaires. Selon les hypothèses que nous avons considérés dans la section 2.2.1, cela indique que le contenu (ou une partie) d'une page d'un ensemble donné correspond à l'information qui permet de contextualiser le contenu d'une ou de plusieurs autres pages de ce même ensemble. Les clusters générés par les méthodes décrites dans la dernière section contiennent des éléments très liés entre eux par la relation de complémentarité.

Nous considérons donc un cluster de pages comme un ensemble d'information auto-descriptif, ce que nous avons appelé dans le chapitre précédent une *unité logique d'information.* La prochaine étape consiste donc à expliciter le contexte associé à cette unité logique d'information. Une fois identifié, nous considérons ce contexte comme étant le contexte de toutes les pages regroupées dans le cluster.

L'explicitation du contexte se fera par l'utilisation d'une *fonction d'abstraction.* Cette fonction reçoit en entrée un cluster de pages et produit en sortie une unité, que nous appelons *nœud contextuel*, contenant l'information contextuelle des pages composant le cluster reçu en entrée. Dans la figure 3.5, considérons le cluster Cl_1 composé des pages p_1, p_3 et p_4. De l'application de la fonction d'abstraction sur le cluster Cl_1 résulte le nœud contextuel $Cont_1$. Ce nœud contient l'information contextuelle des pages composant le cluster duquel il a été généré, i.e. $Cont_1$ contient l'information contextuelle des pages p_1, p_3 et p_4 qui composent le cluster Cl_1. La fonction d'abstraction est appliquée à chacun des clusters générés par l'application de l'algorithme de clusterisation.

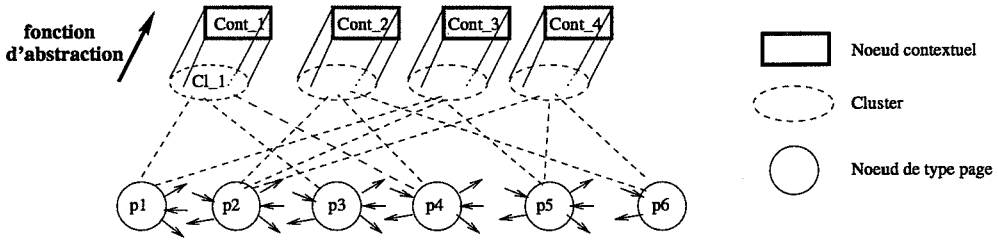

FIG. 3.5: Génération des nœuds contextuels.

À l'instar du contenu des nœuds représentant les pages, les contenus des nœuds contextuels sont aussi représentés à l'aide du modèle vectoriel de Salton. Il faut donc définir la manière dont le contenu d'un nœud contextuel est construit. Bien entendu, le contenu d'un nœud contextuel doit être une fonction du contenu des pages composant son cluster d'origine. Par exemple, le contenu du nœud contextuel $Cont_1$ illustré dans la figure 3.5 doit être une fonction du contenu des pages p_1, p_3 et p_4.

La façon que nous proposons pour calculer le contenu d'un nœud contextuel est basée sur

l'hypothèse suivante : l'information contextuelle d'une page donnée, disons p_i, se trouve la plupart du temps dans le contenu des pages qui lui sont complémentaires et non pas dans son propre contenu. Ainsi, l'information contextuelle d'une page doit se trouver répétée dans un grand nombre d'autres pages appartenant au même cluster qu'elle. Pour capturer cette notion dans le calcul du poids des termes qui représenteront le contenu des nœuds contextuels nous proposons deux formules de pondération pour calculer le poids w_{ik} du terme T_k dans le nœud contextuel dérivé du cluster Cl_i.

La première formule est une formule de type $tf \times idf$ inspirée d'une des formules examinées dans [Sav97, chap. 5]. Ce type de formule est normalement utilisé pour calculer le poids d'un terme à l'intérieur d'un document, le facteur tf[16] correspondant à la fréquence d'occurrence du terme à l'intérieur d'un document et le facteur idf[17] étant une fonction du nombre de documents d'une collection dans lesquels le terme apparaît. Ici, il ne s'agit pas de calculer le poids d'un terme à l'intérieur d'un document mais à l'intérieur d'un cluster de documents. Ainsi, nous allons appeler la formule de pondération que nous décrivons dans la suite ***la formule de la généralisation de*** $tf \times idf$. Le poids w_{ik} du terme T_k dans le nœud contextuel dérivé du cluster Cl_i est calculé selon la formule suivante :

$$w_{ik} = ntf_{ik} \cdot nidf_k \text{ où } ntf_{ik} = \frac{tf_{ik}}{max_z\ tf_{iz}}\ ,\ nidf_k = \frac{idf_k}{\log(n)}\ ,\ idf_k = \log\left(\frac{df_k}{n}\right) \qquad (3.9)$$

où tf_{ik} représente le nombre de pages appartenant au cluster Cl_i dans lesquelles le terme T_k apparaît[18]. Par conséquent, $max_z\ tf_{iz}$ est le nombre maximum de pages du cluster Cl_i ayant au moins un terme en commun. n représente le nombre de nœuds contextuels générés par la fonction d'abstraction et df_k est le nombre de nœuds contextuels dans lesquels le terme T_k apparaît.

Pour ce qui concerne le facteur $nidf_k$ sa fonction est de modérer le poids d'un terme t_k à l'intérieur d'un document lorsque sa fréquence d'occurrence est élevée non seulement dans le document en question mais aussi dans beaucoup d'autres. Dans cette situation, t_k ne constitue pas un bon descripteur du contenu sémantique d'un document car ce terme ne discrimine pas bien le document parmi les autres. Le pouvoir de discrimination d'un terme pourrait ne pas être pris en compte lors de son choix pour représenter le contenu sémantique d'un document, cela dépend de l'utilisation des index des documents pour une application donnée. Il s'avère que dans le contexte d'un système de recherche d'information, le nôtre, il est souhaitable que les termes choisis aient un bon pouvoir de discrimination pour éviter qu'un nombre élevé de documents ne soit retourné en réponse à une requête donnée. Or, dans la prochaine section nous allons décrire comment intégrer le contexte des pages dans leur index. Il est donc souhaitable dans notre contexte que les termes représentant le contexte des pages soit discriminants.

La deuxième formule que nous proposons pour calculer le contenu d'un nœud contextuel à partir du contenu des pages étant à son origine est basée sur un poids seuil calculé en

[16] En anglais : *term frequency.*

[17] En anglais : *inverse document frequency.*

[18] Le terme T_k apparaît dans la page p_i si $w_{ik} > 0$. Notons que nous ne prenons pas en compte les poids des termes qui apparaissent dans les documents groupées dans un cluster. Seul leur présence (ou absence) est considérée.

fonction des poids des termes des pages. Nous allons l'appeler ***la formule du seuil moyen.*** Considérons que nous voulions calculer le contenu du nœud contextuel $Cont_i$ dérivé du cluster Cl_i. D'abord, les vecteurs représentant les contenus des pages de Cl_i sont sommés, notons le vecteur résultant de cette somme $\overrightarrow{V_{int}}$. Ensuite nous calculons le poids seuil qui correspond au poids moyen associé à $\overrightarrow{V_{int}}$, autrement dit, si $\overrightarrow{V_{int}} = (w_1, w_2, \cdots, w_\rho)$, le poids seuil est égal à $(1/l) \times \sum_{k=1}^{\rho} w_k$ où l est le nombre de poids non nuls associés à $\overrightarrow{V_{int}}$. Après, les composantes de $\overrightarrow{V_{int}}$ associées à des valeurs inférieures au seuil calculé sont positionnées à 0. Finalement, la valeur de la composante k, w_k, du vecteur représentant le contenu du nœud contextuel $Cont_i$ est égale à la valeur de cette même composante dans le vecteur $\overrightarrow{V_{int}}$ divisée par le nombre de pages étant à l'origine de $Cont_i$, autrement dit $|Cl_i|$.

3.2.3.1 L'attachement des pages aux contextes

Finalement, nous introduisons une relation entre les pages et les contextes que nous avons appelée *relation d'attachement.* À l'instar de la relation de complémentarité, la relation d'attachement est une relation **floue** dans la mesure où la notion d'intensité de la relation existe. Cependant, contrairement à la relation de complémentarité (section 3.2.1), la relation d'attachement n'est pas une relation binaire (i.e. dans un même ensemble), mais elle porte sur deux ensembles différents : l'ensemble des pages et l'ensemble des contextes générés par la clusterisation des pages.

Considérons Cl_i un cluster généré par l'application de l'une des méthodes de clusterisation décrites dans la section 3.2.2. Supposons aussi que $Cont_i$ est le contexte généré par l'application de la fonction d'abstraction sur le cluster Cl_i. Nous associons une relation d'attachement entre chaque page étant à l'origine du contexte $Cont_i$, i.e. les pages composant Cl_i, et le contexte $Cont_i$.

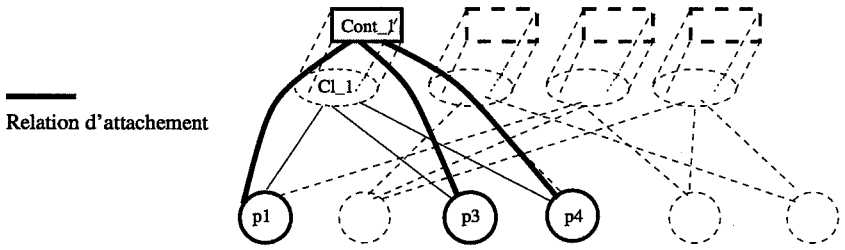

FIG. 3.6: Relation d'attachement entre les pages et leurs contextes correspondants.

Ainsi, si la relation d'attachement existe pour un couple $(p_x, Cont_y)$, cela indique que l'information vehiculée par le contenu de la page p_x trouve son contexte dans l'information véhiculée par le contenu du contexte $Cont_y$.

L'intensité de la relation nous permet de distinguer les pages qui sont plus attachées à un contexte et celles qui le sont moins. Cette notion d'intensité nous permet de capturer l'idée selon laquelle il peut exister des pages plus centrales que d'autres dans un cluster de pages

complémentaires. Une page qui possède une forte complémentarité avec les autres pages dans le même cluster est plus centrale qu'une autre page affichant un degré de complémentarité moins élevé avec les autres pages du cluster. Nous considérons que plus une page est centrale plus elle est *attachée* au contexte qui lui est associé. Ainsi, l'attachement d'une page à un contexte est une fonction de son niveau de complémentarité avec les autres pages formant le cluster duquel le contexte en question a été dérivé. Dans cette première méthode nous calculons l'intensité d'une page à un contexte à l'aide de la formule suivante :

$$Att(p_x, Cont_i) = \frac{1}{|Cl_i|} \cdot \sum_{p_j \in Cl_i} Compl(p_x, p_j) \tag{3.10}$$

où Cl_i est le cluster à l'origine du contexte $Cont_i$ et $Compl(p_i, p_j)$ est la valeur de la complémentarité entre les pages p_i et p_j calculée par l'équation 3.1. Les valeurs des complémentarité entre toutes les paires de pages sont les mêmes que celles utilisées par la méthode de clusterisation dans l'étape précédente.

Nous avons signalé en fin de section 2.2.1 que l'une des applications de l'information contextuelle des pages HTML se trouvait dans l'amélioration de leur index et cela dans le but d'améliorer la qualité des résultats d'un système de recherche d'informations.

L'un des facteurs clés de la performance d'un système de recherche d'information est la qualité de son index. Cette qualité est dépendante de la capacité du module d'indexation à extraire le contenu sémantique d'une page. Nous croyons que l'information contextuelle d'une page est très importante dans la représentation de son contenu sémantique. Cela dit, le fait qu'un moteur de recherche dispose d'index de bonne qualité n'implique pas forcément que leurs résultats seront de bonne qualité. Il faut aussi disposer d'une méthode qui saura exploiter ces index de façon à bien estimer la pertinence des pages aux requêtes. Autrement dit, la qualité de la fonction de correspondance est aussi capitale que la qualité des index pour une bonne performance d'un moteur de recherche. Et ce sera dans la fonction de correspondance que l'intensité de la relation d'attachement entre les pages et leur(s) contexte(s) jouera un rôle important.

Nous avons donc choisi l'application des systèmes de recherche d'information, ou moteurs de recherche, pour démontrer l'utilité de l'information contextuelle des pages. Dans la prochaine section nous suggérons une méthode d'intégration de l'information contextuelle des pages dans le fonctionnement des moteurs de recherche.

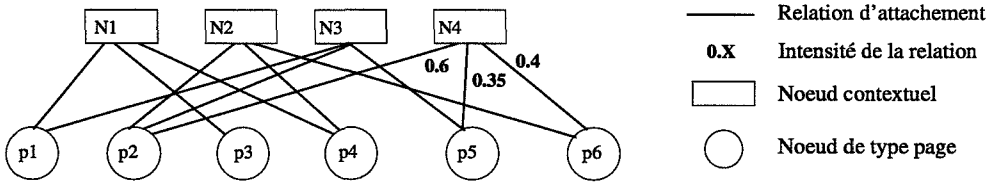

FIG. 3.7: Intensité des attachements entre les pages et leurs contextes correspondants.

3.3 L'intégration de l'information contextuelle des pages dans les moteurs de recherche

Dans la dernière section nous avons suggéré une méthode automatique pour la découverte de l'information contextuelle des pages HTML à l'intérieur d'un site Web. Dans cette section, nous allons montrer l'utilité d'un tel type d'information dans le fonctionnement des moteurs de recherche. Notre objectif ici est d'améliorer la qualité de l'index des pages en y ajoutant l'information contextuelle des pages. En disposant d'index de pages plus précis ou, dans une certaine mesure, *enrichis*, nous espérons pouvoir améliorer l'efficacité du moteur de recherche. Comme on l'a remarqué dans la dernière section un module d'indexation performant, capable d'associer des index de bonne qualité aux pages, ne constitue pas une garantie pour l'efficacité du moteur de recherche. La fonction de correspondance qui estimera la pertinence des documents aux requêtes doit aussi être performante.

Le type de moteur de recherche qui nous intéresse ici est celui associé à un espace de recherche limité[19]. En effet, l'une des limitations du modèle développé dans la dernière section est lié au fait qu'il n'est pas directement utilisable dans une situation où l'espace de recherche aurait une grande dimensionnalité, p.ex. l'espace de recherche correspondant au Web visible entier. Dans la conclusion nous évoquerons cette limitation du modèle tout en proposant des directions vers lesquelles notre modèle pourrait être adapté afin de pouvoir traiter un espace de recherche de grande dimensionnalité.

Dans la suite nous décrivons (i) la manière dont l'information contextuelle peut être intégrée aux index des pages et (ii) la manière dont la fonction de correspondance exploite les index des pages enrichis afin d'améliorer l'estimation de leur pertinence aux requêtes formulées au moteur de recherche.

3.3.1 L'indexation des pages

L'index des pages peut être considéré comme une structure ayant deux niveaux : le premier niveau étant associé à la partie de l'index concernant l'information individuelle de chaque page et le deuxième niveau étant associé à la partie de l'index concernant l'information contextuelle des pages. Les deux niveaux sont reliés par les relations d'attachement décrites précédemment. On peut aussi considérer cette structure comme un graphe biparti dont l'un des sous-ensembles est composé de nœuds représentant les pages et l'autre sous-ensemble est composé de nœuds représentant les contextes des pages. Les arcs pondérés de ce graphe mènent des nœuds représentant les pages vers les nœuds représentant les contextes.

Dans le premier niveau on retrouve l'index du contenu des pages. L'indexation des contenus HTML se fait d'une manière traditionnelle. D'abord nous avons utilisé un *script* simple pour analyser syntaxiquement les contenus afin de supprimer certaines balises du langage HTML avec leur contenu, p.ex. `<APPLET>`, `<STYLE>`, `<SCRIPT>`, `<OBJECT>`, etc. Nous avons aussi ignoré le contenu de l'élément `<META>` du langage HTML. Les raisons de cela sont : (i) le manque de standardisation des valeurs des attributs et de leur sémantique, p.ex. la séman-

[19] L'espace de recherche d'un moteur de recherche correspond à l'ensemble d'unités d'information — des pages HTML, des coordonnées, des notices bibliographiques, etc. — qu'il a indexées. Les réponses d'un moteur sont constituées des pointeurs vers un sous-ensemble d'unités indexées.

tique des valeurs `Description` et `Keywords` pour l'attribut `Content` ne sont pas définies ; (ii) l'utilisation non systématique de cet élément dans les pages HTML ; (iii) l'utilisation courante de cet élément dans un but différent de celui de décrire le réel contenu d'une page.

Après ce nettoyage du contenu des pages HTML nous utilisons un deuxième *script* qui permet d'éliminer les mots-vides du contenu. Les mots-vides — articles, prépositions, etc. — sont des mots réputés pour ne pas être très représentatifs du contenu sémantique d'un document et, par conséquent, ils sont peu utiles dans l'index d'un document. Finalement, sur les termes restants nous appliquons l'algorithme de lemmatisation de Porter [Por97].

À la suite de la lemmatisation nous pouvons finalement créer l'index du contenu des pages. Nous avons utilisé un index du type fichier inversé. Nous avons implémenté les fichiers inversés à l'aide de vecteurs triés [HFBYL92]. Dans le fichier inversé nous ne gardons pas la position d'un terme à l'intérieur dans une page. Seul les poids des termes sont retenus. Le poids d'un terme dans une page HTML est calculé selon l'équation 3.3 déjà décrite dans la section 3.2.1. Toutes ces étapes concernant l'indexation du contenu des pages — analyse syntaxique et nettoyage du contenu des pages, suppression des mots-vides et lemmatisation — sont effectuées avant l'analyse de la complémentarité entre les pages[20].

Dans le deuxième niveau de l'index nous trouvons les index des contextes des pages. Comme nous l'avons expliqué dans la section 3.2.3, les contenus des contextes des pages sont dérivés de l'index des pages. Les index des contextes sont aussi représentés par un index de type fichier inversé. Le fichier inversé est aussi implémenté à l'aide de vecteurs triés. Les poids des termes dans l'index d'un contexte sont calculés selon l'équation 3.9 déjà décrite dans la section 3.2.3.

Finalement, les relations d'attachement entre les pages et les contextes permettent de **relier** les deux niveaux d'index décrits précédemment. Ainsi, l'index général de l'espace d'information considéré est complété par un ensemble de tuples du type :

$$relation\_attachement = (page\_id, contexte\_id, intensité) \tag{3.11}$$

Tel que nous l'avons précisé plus haut cet ensemble de tuples peut être considéré comme un graphe biparti. Les listes d'adjacentes étant une façon traditionnelle de représenter un graphe, nous les avons choisi pour représenter ce troisième élément de notre index qui permet de relier les deux premiers.

On pourrait s'interroger ici sur les raisons de l'existence de deux niveaux d'index plutôt qu'un seul. Pourquoi maintenir dans l'index une séparation de la partie relative au contenu d'une page de celle relative au contexte d'une page? On pourrait par exemple envisager d'intégrer les deux composants dans un seul. Autrement dit, la question soulevée est lié à la représentation du résultat de l'équation ci-dessous qui reflète notre hypothèse de base :

[20] Dans cette section nous n'allons pas rentrer dans les détails de l'implémentation des différents modules/algorithmes d'indexation ni dans les structures de données utilisées. Des explications exhaustives sur les fichiers inversés, les algorithmes de lemmatisation, etc., peuvent être trouvées dans des ouvrages de base sur la recherche d'information [BYRN99] [FBY92] [vR79] [JW97].

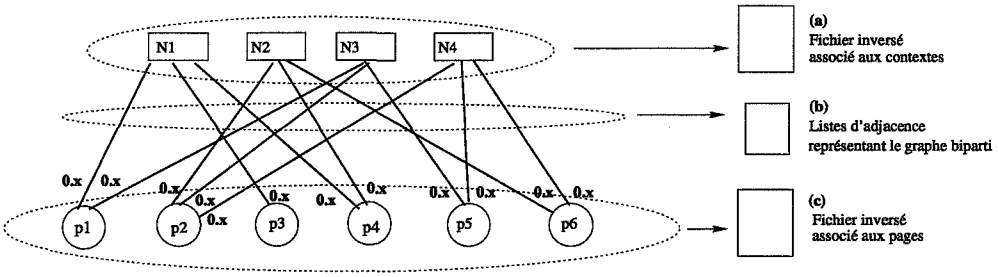

FIG. 3.8: Index associé à l'espace de recherche : (a) fichier inversé associé aux pages (premier niveau) ; (b) listes d'adjacence représentant les relations d'attachement entre les pages et les contextes ; (c) fichier inversé associé aux contextes (deuxième niveau).

$$Index(p_x) = f(Contenu(p_x), Contexte(p_x)) \tag{3.12}$$

Cette équation indique que l'index d'une page est une fonction des termes de son contenu et des termes de son contexte. L'intuition dit que l'on pourrait définir la fonction f de sorte que l'index final de la page p_x soit représenté par un seul composant. Par exemple, nous pourrions rajouter tous ou une partie des termes du contexte de p_x dans l'index déduit seulement du contenu de p_x. Ces termes contextuels pourraient avoir des poids différents dans l'index final de p_x en fonction de l'intensité de la relation d'attachement entre la page p_x et son contexte. Ce type de démarche résulterait dans un index traditionnel à un seul niveau.

Il y a deux raisons pour lesquelles nous avons choisi de garder séparés les deux composants de l'index. La première raison est liée à la situation où une page serait attachée à plus d'un contexte. Si nous combinons deux termes, t_i et t_j, venant de contextes différents nous risquons de créer un index où la combinaison de t_i et t_j pourrait être interprétée comme si le document traitait d'un sujet donnée identifié par la présence de ces deux termes dans un même index. Ces mêmes termes, lorsque considérés séparément, pourraient avoir des sémantiques différentes de celles qu'ils ont lorsqu'ils sont considérés ensemble. Dans le cas où une page est liée à un seul contexte, cette première raison n'est évidement pas fondée.

La deuxième raison est valable aussi bien pour le cas où une page aurait un seul contexte que pour le cas où elle aurait plusieurs contextes. Si nous combinons des termes liés au contenu de la page avec des termes liés au(x) contexte(s) de la page nous perdons la capacité de distinction entre les termes contextuels et ceux non-contextuels. Cette capacité de distinction peut s'avérer utile dans le fonctionnement d'un système de recherche d'information. Comme nous allons le voir dans la prochaine section, nous pouvons identifier deux types différents de termes ayant des fonctions différentes dans l'expression d'un besoin d'information sous la forme d'une requête. Nous croyons pouvoir établir un parallèle entre les deux niveaux de l'index proposés dans cette section avec ces deux différents types de termes ayant des fonctions différentes lorsqu'ils sont utilisés dans l'expression d'un besoin d'information. Nous ferons référence à ces deux types de termes dans la prochaine section.

3.3.2 Le langage des requêtes

Le type d'index que nous avons proposé dans la dernière section est composé de deux niveaux : un premier niveau lié au contenu des pages et un deuxième niveau lié au contenu des contextes des pages, identifié grâce à la méthode d'identification de pages complémentaires décrite dans la section 3.2. Notre idée ici est de proposer un langage de requête capable d'exploiter la distinction entre les parties de l'index concernant chacun des niveaux.

Nous croyons que certains besoins d'informations peuvent être traduits dans des requêtes composées de termes ayant deux fonctionalités différentes. Nous pensons à un premier ensemble de termes qui permettraient de *contextualiser* le besoin d'information et à un deuxième ensemble de termes qui permettraient de focaliser sur le sujet principal du besoin d'information. Par exemple, imaginons un utilisateur interessé par des cours en ligne sur le langage HTML. Son besoin d'information pourrait être traduit dans la requête `cours, langage HTML`. Parmi les termes de cette requête nous pouvons dire que `langage HTML` est *central* au besoin d'information exprimé par la requête alors que le terme `cours` l'est moins. En effet, le terme `cours` contextualise dans un certain sens le terme `langage HTML`.

Imaginons un autre utilisateur intéressé par des informations portant sur les différentes versions du langage HTML. Alors que le sujet central de la requête est toujours le langage HTML, il est question ici de documents portant sur les différentes versions du langage et non plus des cours en ligne du langage. Ce deuxième besoin d'information pourrait être traduit par la requête `version, langage HTML`. Ainsi, le terme `version` contextualise le terme `langage HTML`.

Il nous semble improbable que tous les besoins d'information visant la récupération de documents puissent être traduits par des requêtes composées de termes des deux types cités. En effet, pour certains besoins d'information l'identification ou la distinction des termes appartenant aux deux niveaux peut s'avérer plus facile que pour d'autres besoins d'information. Tout compte fait, le type de requête ou de besoin d'information qui nous intéresse ici est celui pour lequel une distinction des termes des deux types pourrait être faite. Dans [THM+99] trois types de requête différents sont cités dont l'un correspond à celui que nous intéresse. Dans [Dyr98] et [HP93] l'existence et l'utilité d'un type de requête similaire à celui que nous proposons sont évoquées. Dans la conclusion, lors de la discussion sur les limitations de notre modèle nous évoquons les problèmes que peut poser l'utilisation d'un tel langage de requête.

Afin que l'utilisateur puisse distinguer les deux types de termes dans sa requête et ainsi, mieux exprimer son besoin d'information, nous proposons un langage de requête intégrant deux opérateurs : $q_{contexte}$ et q_{sujet}. À l'opérateur $q_{contexte}$ doivent être associés les termes qui contextualisent le besoin d'information alors que les termes centraux à ce même besoin d'information doivent être associés à l'opérateur q_{sujet}. Autrement dit, les termes associés à $q_{contexte}$ affinent ou précisent la description du besoin d'information dont le thème central est associé à q_{sujet}. Ces deux opérateurs permettent à l'utilisateur de différencier le contexte du thème central de son besoin d'information.

Dans le contexte de la fonction de correspondance que nous décrivons dans la prochaine

section, nous faisons l'hypothèse selon laquelle les termes centraux à un besoin d'information (ceux associés à l'opérateur q_{sujet}) se trouveront toujours dans les pages pertinentes à ce besoin. Cependant, les termes qui contextualisent le besoin d'information ne se trouvent pas nécessairement dans ces pages. Ces termes contextuels se trouvent plutôt dans les pages complémentaires aux pages pertinentes. Ainsi, pour estimer la pertinence d'une page à une requête nous allons chercher (i) une parenté entre les termes associés à q_{sujet} et les termes associés à la partie de l'index liée au contenu des pages et (ii) une parenté entre les termes associés à $q_{contexte}$ et les termes associés à la partie de l'index associée au(x) contexte(s) des pages.

3.3.3 La fonction de correspondance

L'efficacité d'un système de recherche d'information est traditionnellement mesurée par les taux de précision et de rappel. En outre, la qualité d'un système doit être définie comme une fonction de non pas un seul facteur, son efficacité, mais d'un ensemble de facteurs tels que l'interface, la rapidité des réponses, la taille de l'index, la fréquence de ses mises à jour, les fonctionalités additionnelles telles que la prise en compte de l'avis de l'utilisateur pour affiner les résultats ou proposer une modification dans la requête[21], la possibilité de demander des pages liées à une page, etc. Pour ce qui est de l'efficacité d'un moteur de recherche, elle ne dépend pas seulement de la qualité de l'index, la capacité d'un système de bien estimer la pertinence des pages aux requêtes d'après les index des pages joue un rôle fondamental dans son efficacité.

La fonction de correspondance est la partie d'un système de recherche d'information qui permet d'estimer la pertinence de chaque page aux requêtes soumises d'après son index. La fonction de correspondance que nous proposons ici s'adapte à l'index à deux niveaux proposé précédemment. Supposons une requête $Q = (q_{contexte}, q_{sujet})$. Afin d'identifier les n pages les plus pertinentes à Q, l'exécution de la fonction de correspondance que nous proposons ici se déroule en trois étapes. Bien entendu, les termes de la requête sont soumis aux mêmes traitements que ceux appliqués aux termes des documents, i.e. le cas échéant, les mots vides éventuellement présents dans les termes de la requête sont éliminés, et ensuite, les termes restants sont lemmatisés. Dans la **première étape** les pages associées soit aux termes du composant q_{sujet} de Q soit aux termes du composant $q_{contexte}$ sont sélectionnées à travers l'analyse de la partie de l'index associée au seul contenu des pages (le niveau inférieur de l'index illustré dans la figure 3.8).

Dans la **deuxième étape**, la fonction de correspondance sélectionne les contextes des pages associés au composant $q_{contexte}$ de Q à travers l'analyse de la partie de l'index concernant les contextes (le niveau supérieur de l'index illustré dans la figure 3.8). La formule qui permet d'évaluer la pertinence du contenu des pages et des contextes aux différents composants de Q dans ces deux premières étapes est l'équation suivante[22] :

$$Sim(D_i, Q) = \sum_{k=1}^{\rho} w_{ik} \cdot w_{qk} \tag{3.13}$$

[21] En anglais : *relevance feedback*.

[22] En réalité cette formule est un produit scalaire de deux vecteurs.

où D_i représente une page ou un contexte si la formule est appliquée dans la première étape ou dans la deuxième étape respectivement (i itère sur les pages et les contextes récupérés dans les deux premières étapes), Q représente le sujet ou le contexte de la requête, w_{ik} représente le poids associé au terme t_k dans le document D_i et w_{qk} représente le poids associé au terme t_k dans la requête Q. Dans la première étape le vecteur Q correspond au vecteur formé par les termes de la requête et les vecteurs D_i correspondent aux index stockés dans le premier niveau de l'index. D'une manière analogue, dans la deuxième étape le vecteur Q correspond au vecteur formé par les termes associés au composant $q_{contexte}$ et les vecteurs D_i correspondent aux index stockés dans le deuxième niveau de l'index.

Dans l'équation 3.13 le poids w_{ik} est calculé selon l'équation 3.3 issue du modèle OKAPI et déjà citée dans la section 3.2.1.1. Pour ce qui est de la pondération des termes de la requête nous utilisons la formule suivante, elle aussi étant issue du modèle OKAPI :

$$w_{qk} = \frac{tf_{qk}}{k_2 + tf_{qk}} \cdot ln\left(\frac{N - df_k}{df_k}\right) \tag{3.14}$$

où tf_{qk} représente la fréquence d'occurrence du terme t_k dans la requête, N représente le nombre de documents de la collection considérée (de pages ou de contextes), df_k représente le nombre de documents de la collection où le terme t_k apparaît et k_2 est une constante qui dans les tests décrits dans le prochain chapitre a pour valeur 1000.

Avant de passer à la troisième étape nous insistons sur le fait que lors de la première étape, non seulement les termes associés à q_{sujet} mais aussi ceux associés à $q_{contexte}$ sont utilisés dans la comparaison avec les index des pages composant le premier niveau de l'index. Ceci parce que le contexte d'une page peut éventuellement se trouver dans la page elle-même. Nous croyons que le contexte d'une page se trouve très souvent ailleurs, i.e. dans des pages complémentaires, mais pas systématiquement. Bien que le caractère hypertextuel du Web favorise la fragmentation d'un document en plusieurs pages, rien n'empêche qu'un document soit rendu dans sa totalité au sein d'une seule page.

Des deux premières étapes résultent deux listes : une liste de pages et une liste de contextes ; les pages et les contextes listés étant ceux que les deux premières étapes de la fonction de correspondance ont jugés comme pertinents aux parties des requêtes auxquelles ils ont été comparés. Étant donné que les pages sélectionnées dans la première étape sont associées chacune à une valeur temporaire de pertinence, nous pouvons imaginer un ordre temporaire des pages selon leurs degrés estimés de pertinence.

Nous pouvons considérer la troisième étape comme celle qui va donner l'estimation finale des pertinences des pages récupérées dans la première étape en fonction aussi de la pertinence de leurs contextes (ceux étant retrouvés dans la deuxième étape) et de l'intensité de leur attachement (des pages) à ces contextes. L'ensemble de relations d'attachement et leur intensité composent le troisième élément de l'index (cf. figure 3.8). En se servant de ces données, notre fonction de correspondance attribuera un degré de pertinence élevée à une page par rapport à une requête si : (i) le contenu de la page est très lié aux termes de la requête et (ii) si la page en question est très attachée à un contexte qui lui, à son tour, doit être très lié aux

termes associés à l'opérateur $q_{contexte}$ de la requête. La degré de pertinence finale d'une page p_x à une requête Q est calculé d'après la formule suivante :

$$R(p_x, Q) = Sim(p_x, q_{sujet} + q_{contexte}) \cdot (\max_{k=1,\cdots,n} (Sim(Cont_k, q_{contexte}) \cdot Att(p_x, Cont_k)) + Sim(p_x, q_{contexte})) \tag{3.15}$$

où $Cont_k$ $(k = 1, 2, \cdots, n)$ représente les n contextes sélectionnés dans la deuxième étape de la fonction de correspondance et $Sim(A, B)$ correspond à la valeur estimée de la pertinence de l'entité A à l'entité B d'après la formule du produit scalaire (équation 3.13). La fonction de correspondance que nous proposons se distingue d'une fonction de correspondance traditionnelle par l'introduction des deuxième et troisième étapes. Quant à la formule d'estimation de pertinence, la différence avec une formule traditionnelle se trouve dans l'introduction du deuxième facteur concernant l'information contextuelle des pages.

Ceci termine la présentation de l'intégration de l'information concernant le contexte des pages dans un moteur de recherche dans le but d'améliorer son efficacité.

Ci-dessous nous proposons un résumé des principales étapes de la mise en œuvre de la première méthode de découverte de l'information contextuelle des pages :

1. nettoyage
2. suppression des mots-vides
3. lemmatisation
4. indexation vectorielle
5. découverte d'unités logiques d'information (section 3.2.2)
6. découverte des contextes des pages (fonction d'abstraction - section 3.2.3)
7. calcul des attachements des pages aux contextes (section 3.2.3.1)

Les quatre premières étapes de *l'algorithme* ci-dessus sont évoquées dans la section 3.3.1. Dans la prochaine section nous décrivons une deuxième méthode de découverte de l'information contextuelle des pages. Contrairement à la méthode décrite dans la section 3.2, la deuxième méthode génère plusieurs niveaux de contextes pour les pages analysées, les contextes s'organisent sous la forme d'une hiérarchie. Par rapport à l'algorithme précédent cette méthode est basée sur l'itération des étapes 4, 5, 6 et 7.

3.4 La hiérarchie des contextes

Dans cette section nous décrivons une deuxième méthode de découverte de l'information contextuelle de pages HTML. Cette deuxième méthode est basée sur celle que nous avons décrite dans la section 3.2, il s'agit en fait d'une extension. Au lieu de créer un seul niveau de contextes pour les pages, nous proposons ici d'en créer plusieurs. Les contextes appartenant aux différents niveaux s'organisent hiérarchiquement. L'idée ici est de découvrir aussi bien des contextes qui sont très spécifiques aux contenus de pages que des contextes plus génériques, ces derniers étant dérivés des premiers. Les contextes les plus spécifiques se localisent aux

niveaux inférieurs de la hiérarchie et les contextes les plus génériques se trouvent aux niveaux supérieurs.

3.4.1 La construction

La construction de la hiérarchie se déroule du bas vers le haut. Le premier niveau de la hiérarchie (le niveau le plus bas) est défini d'une manière très similaire à celle associée à la méthode décrite avant : d'abord un algorithme de clusterisation regroupe les pages complémentaires et, ensuite, une fonction d'abstraction génère les contextes associés aux pages. Pour ce qui concerne l'algorithme de clusterisation rien ne change. Nous utilisons l'un des deux algorithmes décrits dans la section 3.2.2. Cependant, la fonction d'abstraction que nous utilisons dans cette deuxième méthode est légèrement différente de celle associée à la première méthode.

Dans la fonction d'abstraction de la première méthode il n'existe aucune relation entre les nœuds contextuels, i.e. entre les nœuds générés par l'algorithme implémentant la méthode. En revanche, la fonction d'abstraction que nous proposons dans cette deuxième méthode établit des relations entre les nœuds contextuels générés. Ces relations se traduisent par des arcs entres les nouveaux nœuds (figure 3.9). Les types des relations exprimées par ces arcs sont les mêmes que ceux des relations exprimées par les arcs entre les nœuds représentant les pages. Autrement dit, ces arcs peuvent représenter chacun une relation particulière : navigation, structure logique, etc. Encore une fois, nous n'essayons pas d'identifier le type particulier de chacun des liens, ainsi, nous n'attachons pas de type aux relations entre les nœuds. Dans notre méthode, il nous suffit de savoir si entre deux nœuds de telles relations existent ou non.

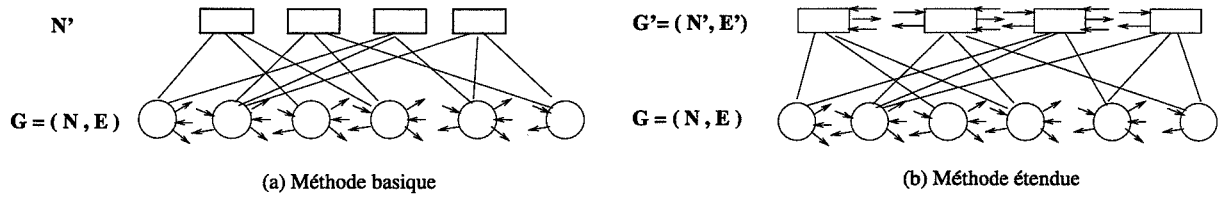

FIG. 3.9: Méthodes basique et étendue.

Étant donné que dans cette deuxième méthode des relations sont établies entre les nouveaux nœuds, nous pouvons associer à cette méthode le processus suivant : à partir d'un graphe $G = (N, E)$, des sous-ensembles de l'ensemble de nœuds N (clusterisation) sont identifiés ; de chacun de ces sous-ensembles, un nouveau nœud est dérivé (fonction d'abstraction) et, finalement, des relations sont établies entre les nouveaux nœuds. Autrement dit, à partir d'un graphe orienté $G = (N, E)$, un nouveau graphe orienté $G' = (N', E')$ est déduit. Le graphe G' peut être considéré comme étant une *généralisation* du graphe G. Plus bas nous expliquons le pourquoi du choix du terme généralisation pour décrire le processus que nous venons de décrire.

Dans les sections 3.2.2 et 3.2.3 nous avons décrit la manière dont les nœuds de N' sont générés à partir des niveaux de complémentarité mesurés entre les paires de nœuds de N. La

fonction d'abstraction définie dans la section 3.2.3 décrit seulement la façon dont le contenu des nouveaux nœuds, i.e. les nœuds composant N', sont définis. Dans la suite, nous définissons comment les relations entre les nœuds de N', i.e. les arcs composant E', sont définis. La fonction d'abstraction utilisée dans la deuxième méthode peut donc être considérée comme une extension de celle utilisée dans la première méthode.

Les relations entre les nœuds contextuels sont établies à partir des relations existant entre les pages composant l'espace de recherche considéré. Autrement dit, les arcs entre les nœuds contextuels (nœuds de G') sont établis en fonction des arcs entre les nœuds représentant les pages (nœuds de G). Ce même type de dépendance se retrouvera dans les niveaux supérieurs de la hiérarchie : les arcs reliant les nœuds situés au niveau $k+1$ de la hiérarchie seront définis à partir des arcs reliant les nœuds situés au niveau k.

Définissons les règles qui régissent la création des arcs entre les nœuds appartenant à un même niveau de la hiérarchie. Considérons deux nœuds D_1 et D_2 situés tous les deux au niveau $k+1$ de la hiérarchie de contextes. Supposons également que D_1 et D_2 aient été générés par l'application de la fonction d'abstraction respectivement sur les clusters C_1 et C_2, tous les deux étant composés de nœuds situés au niveau k de la hiérarchie. Il existe un arc menant de D_1 à D_2 si et seulement s'il existe au moins un arc menant d'un nœud quelconque appartenant à C_1 vers un nœud quelconque appartenant à C_2. Au cas où il existerait plusieurs arcs menant de nœuds de C_1 à nœuds de C_2, un seul arc est créé entre D_1 et D_2. Ainsi, il existe au plus un arc menant d'un nœud, disons N_x, à un autre nœud, disons N_y, tous les deux situés dans un même niveau de la hiérarchie de contextes. On observe donc une cohérence avec le graphe de citations utilisé pour représenter un ensemble de pages reliées entre elles (cf. définition du graphe de citations au début du chapitre 2). Notons qu'un nouvel arc maintient l'orientation de l'arc, ou des arcs, dont il a été dérivé. Dans la figure 3.10 nous illustrons trois situations différentes de connectivité entre nœuds d'un même niveau, disons k, menant tous les trois à une connectivité identique au niveau $k+1$.

Le fait de ne pas distinguer dans le graphe de citations la situation où un seul lien relie deux pages de celle où plus d'un lien relient les mêmes deux pages peut être considéré comme une simplification dans notre modélisation. Il en est de même pour la définition des arcs entre les nœuds contextuels. Dans la figure 3.10(c), les arcs $(p2, p5)$ et $(p3, p4)$ pourraient impliquer la création de deux arcs entre les nœuds contextuels $N1$ et $N2$.

Avec cette nouvelle fonction d'abstraction, extension de celle décrite dans la section 3.2.3, nous disposons de l'outil de base pour la construction de la hiérarchie de contextes. Cet outil de base correspond à la procédure qui permet de générer le premier niveau de la hiérarchie de contextes à partir du graphe de nœuds représentant les pages tout en définissant des relations entre les nœuds du premier niveau de la hiérarchie. Comme nous l'avons dit plus haut, cette procédure peut être considérée comme l'application d'un algorithme qui reçoit en entrée un graphe $G = (N, E)$ et génère en sortie un graphe $G' = (N', E')$ où $|N'| < |N|$. La génération des niveaux supérieurs de la hiérarchie est obtenue à travers l'itération de la procédure de base un certain nombre de fois. De chaque itération résulte un niveau qui est additionné au sommet de la hiérarchie. L'itération k reçoit en entrée le graphe genéré dans l'itération précédente, i.e. l'itération $k-1$, et génère le niveau k de la hiérarchie. L'algorithme de construction

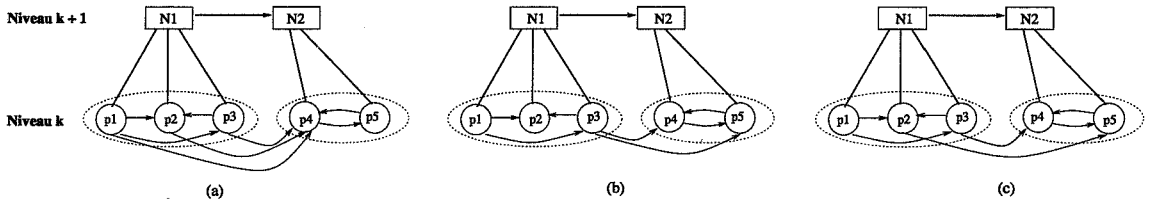

FIG. 3.10: Trois exemples de connectivité entre les clusters de nœuds d'un même niveau : (a) connectivité $n \times 1$ (tous les nœuds d'un cluster pointant sur un même nœud d'un deuxième cluster) ; (b) connectivité $1 \times n$ (un nœud d'un cluster pointant sur tous les nœuds d'un deuxième cluster) ; (c) connectivité $m \times n$.

de la hiérarchie se résume donc dans plusieurs itérations de la procédure de base constituée des trois étapes suivantes exécutées successivement : calcul de complémentarité; clusterisation ; application de la fonction d'abstraction (figure 3.11).

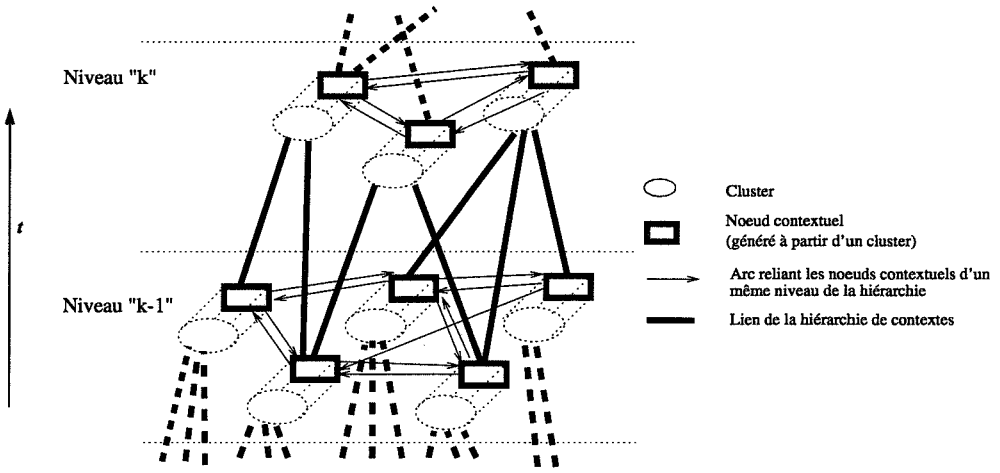

FIG. 3.11: Niveaux consécutifs de la hiérarchie de contextes.

Par rapport au nombre d'itérations que l'algorithme doit exécuter, nous avons utilisé dans les tests que nous décrivons dans le prochain chapitre le critère suivant : l'exécution doit se terminer soit lorsqu'un niveau avec un seul nœud est généré soit un nombre maximal d'itérations prédéfini est atteint. D'autres critères d'arrêt basés par exemple sur la complémentarité moyenne entre les nœuds contextuels du niveau le plus supérieur de la hiérarchie pourraient aussi être conçus. L'algorithme de construction de la hiérarchie de contextes est résumé ci-dessous :

- Entrées : (i) un graphe orienté G = (N,E) où N est l'ensemble de nœuds et E l'ensemble d'arcs et (ii) la valeur n qui correspond au nombre maxi-

mal de niveaux de la hiérarchie à obtenir à en sortie de l'algorithme.

- Soit *Hierarchie* la hiérarchie de contextes, *niv* le nombre de niveaux de la hiérarchie.
- $k = 0$;
- Tant que $k < n$ ET $|N| > 1$:
  - Calculer la complémentarité $Compl(N_i, N_j)$ pour tous les $N_i, N_j \in N$;
  - Clusteriser les nœuds de N en se basant sur l'intensité de leur complémentarité deux à deux ;
  - Appliquer la fonction d'abstraction à tous les clusters générés dans l'étape précédente. De cela résulte le graphe $G' = (N', E')$;
  - Faire de G' un nouveau niveau de la hiérarchie en le rajoutant dans le haut de la hiérarchie. La liaison entre les deux niveaux est faite par les relations d'attachement existantes entre les nœuds de N et N' ;
  - k = k + 1 ;
  - G = G' ;

Dans la section 3.2.3.1, nous avons introduit les relations d'attachement entre les pages et les contextes. Dans la hiérarchie de contextes, nous trouvons le même type de relation excepté le fait qu'il s'agira d'un attachement entre deux contextes appartenant à deux niveaux consécutifs de la hiérarchie et non pas d'un attachement entre une page et un contexte. L'interprétation de la relation d'attachement entre deux contextes est analogue à celle existante entre une page et un contexte. Supposons qu'un contexte, disons, $Cont_i$, situé dans le niveau $k-1$ de la hiérarchie soit attaché à un contexte, disons $Cont_j$, situé au niveau k. Ainsi, le contenu de $Cont_i$ trouve son contexte dans l'information véhiculée par le contenu de $Cont_j$. Le fait que $Cont_i$ représente dejà un nœud contextuel peut rendre difficile l'interprétation de la relation de l'attachement entre les deux nœuds contextuels. Cependant, si nous abstrayons le type d'information vehiculéee par $Cont_i$ et nous le considérons comme un nœud porteur d'une information *ordinaire* — comme celle véhiculée par une page HTML — , l'interprétation de la relation d'attachement entre les deux nœuds devient plus simple.

Une autre façon d'interpréter la relation d'attachement entre $Cont_j$ et $Cont_i$ consiste à considérer l'information vehiculée par $Cont_j$ comme étant une généralisation de l'information vehiculée par $Cont_i$. Cela est possible si nous considérons que le contexte d'une information représente dans une certaine mesure une généralisation de cette information. Ainsi, si $Cont_j$ constitue le contexte de $Cont_i$, nous pouvons voir son contenu comme étant une généralisation du contenu de $Cont_i$. D'une manière analogue nous pouvons considérer les informations vehiculées par les nœuds du niveau $k+1$ de la hiérarchie comme étant une généralisation de l'information véhiculée par les nœuds situés au niveau k.

L'intensité de l'attachement entre deux contextes est calculée d'une façon similaire à celle utilisée pour calculer l'attachement entre une page et un contexte. Il s'agit en fait d'une généralisation de l'équation 3.10 introduite dans la section 3.2.3.1. Considérons les nœuds contextuels $Cont_i$ et $Cont_j$ situés aux niveaux $k-1$ et k respectivement de la hiérarchie. L'intensité de la relation d'attachement entre $Cont_i$ et $Cont_j$ est calculée par l'équation suivante :

$$Att(Cont_i, Cont_j) = \frac{1}{|Cl_j|} \cdot \sum_{N_x \in Cl_j} Compl(Cont_i, N_x) \tag{3.16}$$

où Cl_j est le cluster étant à l'origine du contexte $Cont_j$ et $Compl(Cont_i, N_x)$ est l'intensité de la complémentarité entre les nœuds $Cont_i$ et N_x calculée par l'équation 3.1. Notons que les nœuds $Cont_i$ et N_x appartiennent à un même niveau de la hiérarchie de contextes, en l'occurrence le niveau $k-1$.

3.4.2 Les applications de la hiérarchie de contextes

3.4.2.1 L'indexation de sites

Nous pouvons imaginer plusieurs applications pour la hiérarchie de contextes associée à un ensemble cohérent de pages, c.-à-d. à un site tel que nous l'avons défini dans la section 3.1. La hiérarchie (ou une partie) peut par exemple être utilisée pour représenter ou résumer un site. Autrement dit, on peut se servir de la hiérarchie pour créer l'index d'un site. Par exemple, nous pouvons imaginer une manière capable d'identifier un niveau particulier de la hiérarchie qui soit représentatif du contenu du site. L'index du site serait donc modélisé par un ensemble *d'entités* associées chacune au contenu d'un nœud contextuel faisant partie du niveau sélectionné (figure 3.12).

Nous pouvons aussi imaginer une représentation plus traditionnelle pour l'index du site, p.ex. celle basée sur le modèle vectoriel où un *document* est associé à un vecteur de termes. Dans notre cas le document serait le site entier. Afin d'utiliser cette représentation pour l'index d'un site nous pourrions par exemple calculer un vecteur centroïde des vecteurs associés aux nœuds contextuels composant le niveau sélectionné de la hiérarchie. Un exemple est proposé dans la figure 3.12. Nous n'y avons pas illustré les arcs reliant les nœuds contextuels d'un même niveau de la hiérarchie pour une question de simplicité. Cependant, nous pouvons aussi imaginer la prise en compte des arcs entre les nœuds contextuels d'un niveau de la hiérarchie dans le calcul du vecteur centroïde.

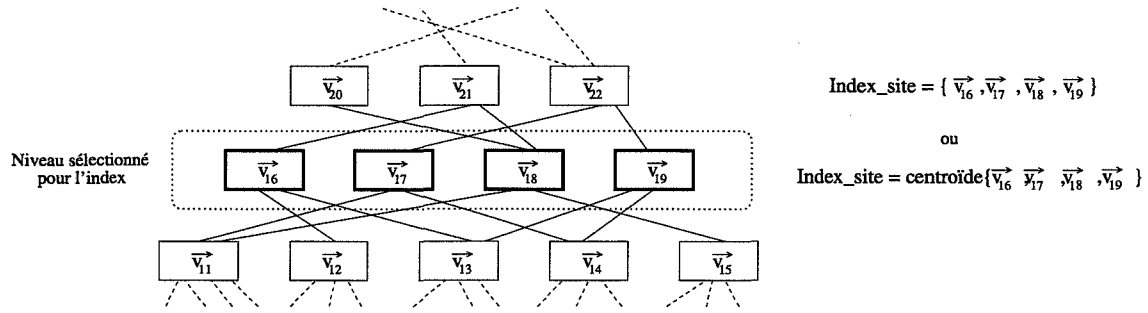

FIG. 3.12: Utilisant la hiérarchie de contextes pour indexer un site.

La représentation du Web par un ensemble de sites plutôt que par un ensemble de pages peut être intéressante dans l'objectif de réduction de la dimensionalité du Web. La granularité

page peut s'avérer trop fine pour certaines applications, une granularité moins fine peut être plus adéquate dans certains contextes. C'est dans cette direction que la fonctionalité de recherche des sites a été intégrée au moteur de recherche *Caloga* (`http://fr.web.caloga.com/`).

3.4.2.2 La navigation à l'intérieur d'un site

Une deuxième application de la hiérarchie de contextes serait d'être à la base d'un outil d'aide à la navigation dans un site. Nous pouvons imaginer un espace de navigation à deux niveaux, un niveau inférieur et un autre supérieur, que nous allons appeler dans la suite N_{inf} et N_{sup} respectivement. N_{inf} serait composé d'entités représentant une information de granularité très fine, nous pensons plus précisément aux pages d'un site. N_{inf} serait donc constitué du graphe de pages d'un site. Les arcs entre les pages représenteraient la multitude de relations qu'un lien hypertexte peut représenter. En revanche N_{sup} contiendrait des entités représentant une information de granularité moins fine, nous pensons aux contextes des pages. N_{sup} serait constitué par le graphe associé à la hiérarchie de contextes. Notons que les arêtes de ce dernier graphe sont uniquement celles existant entre les nœuds contextuels de deux niveaux successifs de la hiérarchie de contextes, ces arêtes sont dessinées en gras dans la figure 3.11. Ces arêtes reliant les nœuds contextuels représenteraient les relations de généralité/spécificité associées aux attachements évoqués plus haut. Enfin, N_{inf} et N_{sup} seraient reliés par les relations d'attachement existantes entre les pages et les nœuds contextuels appartenant au premier niveau de la hiérarchie de contextes.

Finalement, afin que l'utilisateur puisse passer du niveau inférieur au niveau supérieur et vice versa, des relations d'attachement peuvent être introduites à travers des arcs reliant les nœuds représentant les pages (niveau inférieur) et les nœuds représentant les contextes (niveau supérieur). Comme nous l'avons vu dans la section 3.2.3.1, les pages sont directement attachés aux nœuds contextuels composant le premier niveau de la hiérarchie de contextes (cf. figure 3.7). En considérant la propriété de transitivité de la relation d'attachement, les pages sont indirectement attachées aux contextes situés aux niveaux supérieurs de la hiérarchie. Nous pourrions définir une façon de calculer l'intensité de l'attachement entre une page et des contextes situés aux niveaux supérieurs de la hiérarchie (cf. équation 3.20) et selon l'intensité de l'attachement nous créerions ou non un arc reliant une page à un contexte dans le graphe sur lequel est basé l'outil de navigation que nous proposons ici. Une alternative consiste à répercuter dans le graphe uniquement les relations d'attachement entre les pages et les nœuds contextuels du premier niveau de la hiérarchie (figure 3.13).

En termes d'itération avec l'outil de navigation, l'utilisateur pourrait naviguer aussi bien dans le niveau inférieur que dans le niveau supérieur. Afin de passer d'un niveau à l'autre, l'utilisateur se servirait des liens associés aux relations d'attachement entre les pages et les contextes. Une navigation type dans le graphe représenté par l'espace à deux niveaux que nous proposons se déroulerait la façon suivante. L'utilisateur commencerait par exploiter le niveau supérieur en navigant parmi les contextes du site ; cela lui permettrait d'appréhender l'information véhiculée par le site et d'identifier un contexte étroitement associé à son besoin d'information. Une fois le contexte identifié, l'utilisateur pourrait *descendre* au niveau inférieur pour accéder à(aux) la page(s) attachée(s) au contexte sélectionné. Ces pages serviraient donc de points d'entrée aux parties du site susceptibles d'être pertinentes au besoin

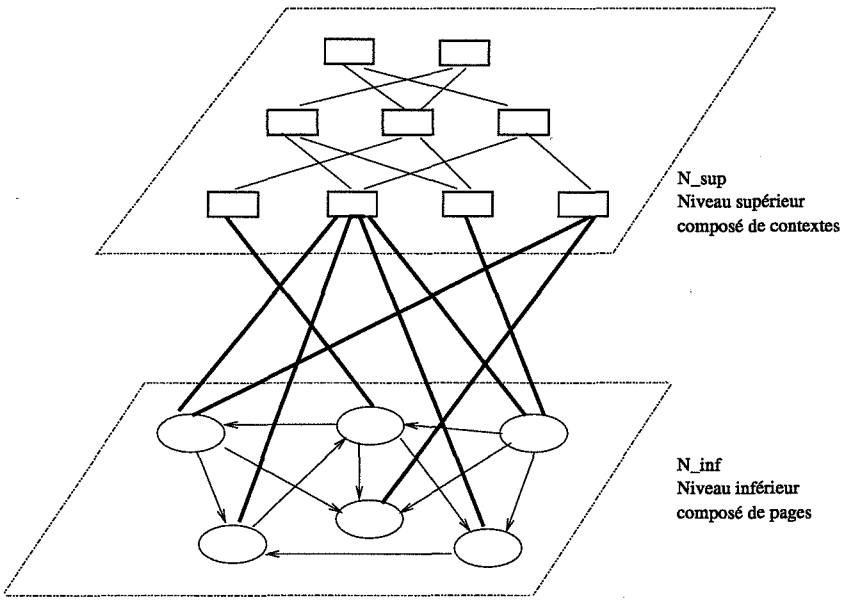

FIG. 3.13: Navigation dans l'espace de recherche à l'aide de la hiérarchie de contextes.

d'information de l'utilisateur. L'utilisateur pourrait donc exploiter le voisinage de ces pages et éventuellement remonter au niveau supérieur (niveau contextuel) pour continuer sa navigation à un niveau d'information plus général. La navigation de l'utilisateur dans un espace de dimension réduite et contenant une information plus généralisée peut atténuer les problèmes cognitifs inhérents au processus de navigation que nous avons évoqués dans la section 1.3.3.

D'autres travaux liés à la récupération d'information dans un système hypertexte ont proposé une structure à deux niveaux pour représenter l'information contenue dans le système [BV92] [CK97] [ACG91]. Dans le niveau supérieur on retrouve traditionnellement un type de connaissance qui décrit d'une manière alternative l'information située au niveau inférieur. Le processus de récupération d'information consiste à naviguer dans chaque niveau et entre les niveaux selon la manière que l'utilisateur trouve la plus adéquate ou la plus simple pour appréhender l'information disponible.

3.4.2.3 Moteurs de recherche

Finalement, la troisième application de la hiérarchie de contextes que nous suggérons est celle liée au fonctionnement d'un moteur de recherche. Dans la section 3.3 nous avons montré l'utilité d'un niveau de contextes dans l'indexation des pages. Ici, nous allons essayer de voir l'utilité de plusieurs niveaux de contextes et non pas d'un seul. L'idée est d'intégrer dans l'index d'une page non seulement son contexte le plus proche (celui appartenant au premier niveau de la hiérarchie de contextes) mais aussi ses contextes plus lointains ou génériques que son contexte le plus proche.

Supposons une requête formulée selon le langage de requêtes que nous avons introduit dans la section 3.3.2, c.-à-d. une requête où le contexte du besoin d'information est distingué du

sujet central. En plus, imaginons que dans la collection de pages il existe une page très pertinente à la requête sans qu'aucun de ces deux contextes les plus proches, $Cont_i$ et $Cont_j$, ne soit très pertinent au contexte spécifié dans la requête (figure 3.14). Dans cette situation, un système basé sur un seul niveau de contextes sous-estimera la pertinence de la page en question.

Imaginons maintenant un système disposant d'une hiérarchie de contextes et que, dans le deuxième niveau de cette hiérarchie, il existe un troisième contexte associé à la page que nous avons signalée avant comme étant très pertinente à la requête. De par la méthode de construction de la hiérarchie, ce troisième contexte a été dérivé d'un ensemble de contextes appartenant au premier niveau dans lequel nous trouvons les deux contextes précédemment cités. Il peut s'avérer que ce troisième contexte soit très lié au contexte spécifié dans la requête de l'utilisateur. Ainsi, si le système en question dispose de la hiérarchie de contextes, il sera capable de retrouver ce troisième contexte de la page — celle que nous savons être pertinente à la requête soumise — et ne plus sous-estimer sa pertinence à la requête.

En bref, nous supposons que le degré de spécificité/généralité du contexte spécifié par l'utilisateur dans sa requête n'est pas toujours équivalent au degré de spécificité/généralité d'un contexte situé au premier niveau d'une page pertinente. Les niveaux supérieurs pallient l'écart en termes de spécificité/généralité pouvant exister entre le contexte spécifié par l'utilisateur dans sa requête et les contextes situés au premier niveau de la hiérarchie de contexte.

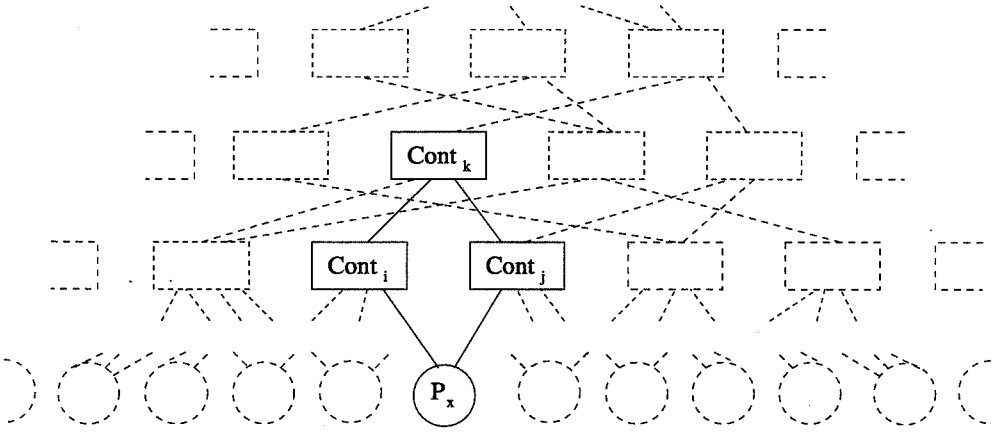

FIG. 3.14: Fusion de deux contextes dans un autre plus général.

Compte tenu que notre objectif dans ce chapitre est de proposer un modèle de recherche d'information pour un système hypertexte, il est évident que, parmi les trois applications de la hiérarchie de contexte que nous venons de citer, nous avons choisi les moteurs de recherche pour étudier l'utilité de la hiérarchie. À l'image de ce que nous avons fait pour le cas où l'information contextuelle découverte est limitée à un seul niveau de nœuds contextuels (cf. section 3.3), nous décrivons dans la section suivante l'intégration de la hiérarchie de contextes dans le fonctionnement d'un moteur de recherche.

3.5 L'intégration de la hiérarchie de contextes dans les moteurs de recherche

Dans cette section nous détaillons l'intégration de l'information contextuelle représentée par la hiérarchie de contextes dans le fonctionnement d'un moteur de recherche. L'intégration est très similaire à celle que nous avons décrite dans la section 3.3 où l'information contextuelle était constitué d'un seul niveau. Le but de cette intégration est évidemment le même qu'avant, c.-à-d. celui d'améliorer la qualité des index des pages de sorte qu'ils représentent bien l'information qu'elles véhiculent. Un index de bonne qualité étant nécessaire mais pas suffisant pour une bonne efficacité d'un moteur de recherche, nous introduisons aussi une fonction de correspondance qui exploite l'information condensée dans l'hiérarchie de contextes de sorte à mieux estimer la pertinence des pages aux requêtes.

Dans la suite nous décrivons (i) la structure de l'index intégrant la hiérarchie de contextes et (ii) la fonction de correspondance associée. Pour ce qui est du langage de requête utilisé par le modèle, il est identique à celui que nous avons décrit dans la section 3.3.2.

3.5.1 L'indexation des pages

À l'instar de l'index décrit dans la section 3.3.1, l'index des pages est aussi une structure à deux niveaux : le premier niveau étant associé aux pages et le deuxième étant associé aux contextes. La seule différence par rapport à l'index associé à la première méthode est que dans le deuxième niveau nous retrouvons les contextes organisés dans une hiérarchie et non plus dans une structure plate.

En ce qui concerne le premier niveau, les contenus des pages y sont indexés de la même façon que celle décrite dans la section 3.3.1. Ainsi, avant l'attribution des poids aux termes, il est question d'une étape de nettoyage du contenu des pages, de suppression de mots-vides et finalement de lemmatisation. Finalement, les termes restants sont associés à des poids calculés selon l'équation 3.3, que nous avons introduite dans la section 3.2.1.

Pour ce qui est du deuxième niveau de l'index, au lieu d'être associé à un ensemble de contextes organisés d'une manière plate, nous retrouvons ici un ensemble de contextes structurés dans une hiérarchie. Ainsi, non seulement le contenu des contextes doit être indexé mais la structure de la hiérarchie doit être aussi sauvegardée. En ce qui concerne la pondération des termes composant le contenu des contextes, le calcul de leur poids est fait par l'équation 3.9 à l'instar de la pondération utilisée dans la première méthode.

La structure du graphe sous-jacent à la hiérarchie de contextes, c.-à-d. les arcs reliant les nœuds contextuels situés à des niveaux consécutifs de la hiérarchie, représentent les relations d'attachement entre les contextes. Étant donné que les arcs entre les pages et les contextes du premier niveau représentent aussi des relations d'attachement, la structure induite par les deux derniers ensembles d'arcs représentant des relations d'attachement est gardée conjointement dans l'index. La structure finale (figure 3.15) décrivant les relations entre les pages

et les contextes est celle d'un graphe non-orienté aux arcs pondérés. Cependant, ce graphe non-orienté pondéré ne constitue plus un graphe biparti comme c'était le cas dans l'index associé à la première méthode.

Le tuple représentant les relations d'attachement se généralise : les relations ne relient plus seulement des pages à des contextes mais aussi des contextes entre eux. Ainsi, le dernier composant de l'index qui permet de relier les deux premiers niveaux est constitué d'un ensemble de tuples du type :

$$relation\_attachement = (contexte\_id, contexte\_id, intensité) \tag{3.17}$$

Finalement, il est important de souligner le fait que les arcs reliant des nœuds contextuels situés à un même niveau de la hiérarchie de contextes ne sont pas sauvegardés dans l'index. Bien que ces arcs soient fondamentaux pour la construction de la hiérarchie, ils ne sont pas utilisés par la fonction de correspondance que nous décrivons dans la prochaine section.

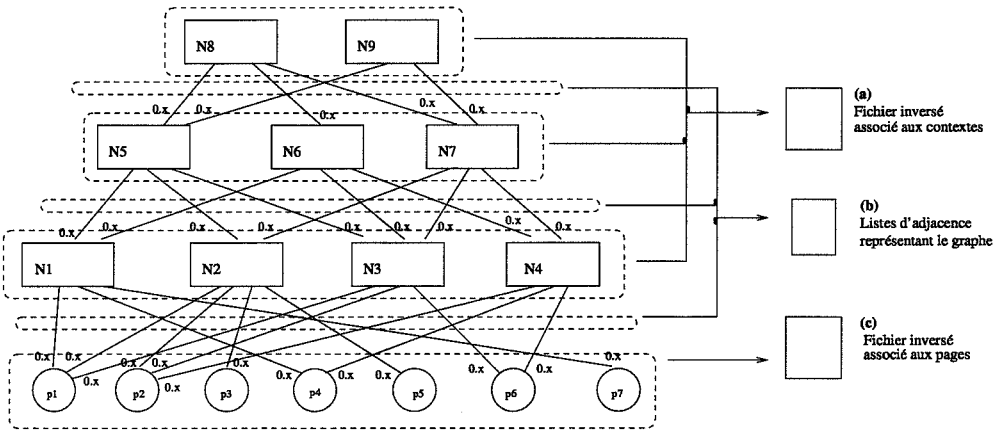

FIG. 3.15: Index associé à l'espace de recherche constitué par les pages d'un site : (a) fichier inversé associé aux pages (premier niveau) ; (b) listes d'ajacence représentant les relations d'attachement entre les pages et les contextes et entre les contextes ; (c) fichier inversé associé à la hiérarchie de contextes (deuxième niveau).

3.5.2 La fonction de correspondance

À l'image de l'indexation, la fonction de correspondance associée à cette deuxième méthode de découverte de l'information contextuelle est aussi très similaire à sa correspondante utilisée dans la première méthode. Il s'agit d'une extension de la fonction décrite dans la section 3.3.3. La seule différence entre les deux est liée à la manière dont la valeur $Att(p, C_k)$, présente dans l'équation 3.15, est calculée.

Le calcul de la fonction de correspondance que nous proposons pour cette deuxième méthode est composée exactement des mêmes trois étapes que celles proposées pour la première

méthode. La **première étape** consiste à identifier les pages liées aux termes associés aux deux composants de la requête, q_{sujet} et $q_{contexte}$. Pour cela, les index contenus dans la partie de l'index associé au contenu des pages (niveau inférieur de l'index illustré dans la figure 3.15) sont analysés. La fonction de correspondance utilisée dans cette première étape est le produit scalaire (équation 3.13) que nous avons défini dans la section 3.3.3.

La **deuxième étape** identifie les contextes liées aux termes associées au composant $q_{contexte}$ de la requête. Elle est exécutée de manière identique à sa correspondante de la première méthode. Ainsi, la pertinence des index représentant les nœuds contextuels de la hiérarchie (index situés dans la partie supérieure de l'index illustré dans la figure 3.15) au contexte de la requête, $q_{contexte}$, est aussi évaluée à l'aide du produit scalaire (équation 3.13). La seule petite différence dans le déroulement de cette deuxième étape par rapport à celui de sa correspondante de la première méthode est que, dans le cas présent, le contexte de la requête est comparé à un nombre supérieur de contextes. En effet, il ne s'agit plus d'analyser les index associés aux nœuds d'un seul niveau, mais il faut analyser les index associés à tous les nœuds composant la hiérarchie de contextes. De ces deux premières étapes résultent une liste de pages et une liste de contextes.

La réelle différence entre la fonction de correspondance que nous décrivons dans cette section avec celle décrite dans la section 3.3.3 se trouve dans la **troisième étape**. Comme nous le savons déjà, la troisième étape est chargée d'estimer la pertinence finale des pages récupérées dans la première étape selon les contextes récupérés dans la deuxième étape. Ces estimations sont basées sur l'intensité de l'attachement entre les pages et les contextes présents dans les listes retournées par les deux premières étapes.

Dans le contexte de la première méthode nous avons défini comment calculer l'intensité de l'attachement entre une page et un contexte (cf. équation 3.10). Dans le contexte de cette deuxième méthode, nous avons généralisé cette dernière formule de façon à pouvoir calculer non seulement l'intensité de l'attachement entre les pages et les contextes situés au premier niveau de la hiérarchie mais aussi l'intensité de l'attachement entre deux nœuds contextuels situés à deux niveaux consécutifs quelconques de la hiérarchie de contextes (cf. équation 3.16)[23].

Comme la deuxième étape peut récupérer des contextes situés à n'importe quel niveau de la hiérarchie et non pas seulement parmi ceux situés au premier niveau, il nous faut encore généraliser la formule d'attachement. Nous avons besoin d'une fonction qui permette de calculer l'attachement d'une page à un contexte situé à un niveau autre que le premier niveau de la hiérarchie. Pour cela nous prolongeons par transitivité la relation d'attachement que nous avons définie dans la section 3.2.3.1. Ainsi, si une entitié n_i est attachée à une autre entité n_j et si n_j est à son tour attachée à une troisième entité n_k, alors n_i est aussi attachée à n_k. Or, ni l'équation 3.10 ni l'équation 3.16 ne permettent de calculer l'intensité de ce nouvel attachement. Dans la suite, nous allons proposer une formule pour effectuer ce calcul. Cette formule s'inspirera des deux dernières formules citées.

[23]Remarquons que dans le contexte de la première méthode, l'équation 3.10 permet de calculer aussi en quelque sorte l'intensité de l'attachement entre deux entités (une page et un contexte) situés dans deux niveaux consécutifs et étroitement liés vu que les entités de l'un des niveaux sont directement dérivées des entités de l'autre (cf. figure 3.7).

Considérons l'exemple illustré dans la figure 3.16. Si la première étape a récupéré la page p_4 et la deuxième étape a récupéré le contexte D_5, pour calculer la pertinence finale de p_4 à la requête il faut connaître l'intensité de l'attachement de p_4 à D_5. Or, l'équation 3.10 nous permet de connaître l'intensité de l'attachement de p_4 à D_1, $Att(p_4, D_1)$, et l'équation 3.16 nous permet de connaître de l'intensité de l'attachement de D_1 à D_5, $Att(D_1, D_5)$. Toutefois, aucune des deux formules ne nous fournit ce qui nous intéresse réellement, à savoir l'attachement de p_4 à D_5, $Att(p_4, D_5)$. Intuitivement, $Att(p_4, D_5)$ doit être une fonction de $Att(p_4, D_1)$ et de $Att(D_1, D_5)$.

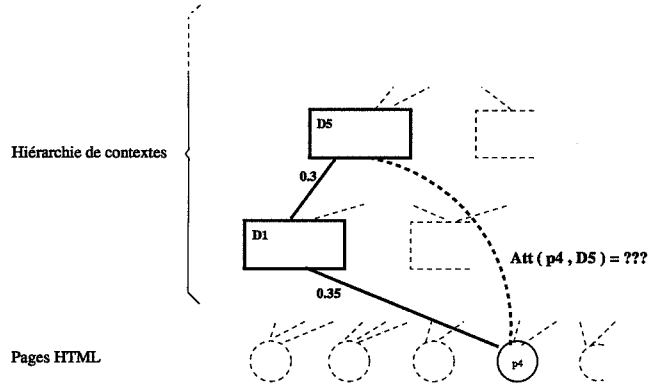

FIG. 3.16: L'intensité de l'attachement entre deux entités situées à deux niveaux non-successifs est inconnue.

Pour simplifier l'explication, considérons que les pages se trouvent au niveau 0 de la hiérarchie. Reprenons la hiérarchie de contextes et l'ensemble de pages illustrés dans la figure 3.15. Dans la suite, nous allons considérer que l'ensemble de pages constituera un niveau additionnel de la hiérarchie de contextes et, qu'ainsi, la hiérarchie illustrée dans la figure 3.15 est composé de 4 niveaux au lieu de 3 niveaux. Nous devons donc définir une fonction qui permette de calculer l'intensité entre deux entités situées à deux niveaux non successifs de la hiérarchie de contextes.

Dans l'explication qui suit nous allons utiliser le graphe non-orienté sous-jacent à la hiérarchie où un nœud représente une page ou un contexte et un arc représente une relation d'attachement entre une page et un contexte ou entre deux contextes différents.

Définissons une première mesure associée aux chaînes de ce graphe. Les chaînes qui nous intéressent ici sont un peu particulières dans la mesure où elles doivent comprendre au plus un nœud de chaque niveau de la hiérarchie représentée par le graphe. Nous appellerons *chaînes monotones* les chaînes vérifiant cette propriété. Ainsi, sur le graphe de la figure 3.15, alors que la chaîne $p_1 \rightarrow N_1 \rightarrow N_5$ vérifie la propriété que nous venons de citer, la chaîne $p_1 \rightarrow N_1 \rightarrow N_5 \rightarrow N_2$ ne la vérifie pas puisqu'elle contient deux nœuds, N_1 et N_2, qui appar-

tiennent à un même niveau de la hiérarchie représentée par le graphe.

Nous définissons la **raideur** d'une chaîne reliant deux nœuds du graphe, disons $ch_i = n_1 \rightarrow \cdots \rightarrow n_j \rightarrow \cdots \rightarrow n_k$, et vérifiant la propriété que nous venons de citer comme la moyenne des poids des arêtes composant la chaîne. Le poids d'une arête du graphe correspond à l'intensité de la relation d'attachement que l'arête représente. Ainsi la raideur de ch_i, $Raideur(ch_i)$ se calcule par :

$$Raideur(ch_i) = \frac{1}{long(ch_i)} \times \sum_{l_{ab} \in ch_i} poids(l_{ab}) \tag{3.18}$$

où l_{ab} représente une arête entre les nœuds n_a et n_b dans le graphe, $poids(l_{ab})$ représente le poids associé à cette arête et $long(ch_i)$ représente la longueur (le nombre d'arcs) de la chaîne ch_i.

Définissons maintenant l'intensité de l'attachement entre deux entités[24] représentées par deux nœuds du graphe, disons N_x et N_y, reliés par une chaîne monotone, disons ch_k, de longueur supérieure à 1[25]. Nous définissons que l'intensité de cet attachement comme la raideur de la chaîne ch_k. Autrement dit :

$$Att(N_x, N_y, ch_k) = Raideur(ch_k) \tag{3.19}$$

où $Att(N_x, N_y, ch_k)$ correspond à l'intensité de l'attachement entre les entités représentées par N_x et N_y suivant la chaîne ch_k.

Or, les nœuds représentant deux entités situées à des niveaux non-consécutifs de la hiérarchie peuvent être reliés par plusieurs et non pas une seule chaîne monotone dans le graphe sous-jacent à cette hiérarchie[26].

L'intensité de l'attachement entre une page et un contexte calculé par l'équation 3.19 est fonction d'une chaîne qui les relie. Puisqu'il peut y avoir plusieurs chaînes reliant les nœuds représentant une page et un contexte dans le graphe sous-jacent à la hiérarchie, selon la chaîne considérée, l'équation 3.19 peut fournir des intensités différentes pour l'attachement entre une page et un contexte. Il nous faut donc définir l'intensité de l'attachement entre une page et un contexte quel que soit le nombre de chaînes qui les relie. Dans la suite, on appellera ce type d'attachement *attachement global.*

Nous définissons l'intensité de l'*attachement global* entre deux nœuds appartenant à deux niveaux non-successifs de la hiérarchie représentée par le graphe comme étant la raideur de la chaîne la plus raide reliant ces deux nœuds. Cette intensité est donc calculée par l'équation suivante :

$$Att(N_x, N_y) = \max_{i=1,2,\cdots,m} (Att(N_x, N_y, ch_i)) \tag{3.20}$$

[24] L'attachement entre une page et un contexte ou l'attachement entre deux contextes différents.

[25] La longueur d'une chaîne est son nombre d'arêtes.

[26] Plusieurs chaînes peuvent relier un nœud représentant une page et un autre nœud représentant un contexte lorsque l'algorithme de clusterisation utilisé pour regrouper des entités complémentaires (cf. section 3.2.2) génère des clusters non-disjoints.

où m correspond au nombre de chaînes monotones reliant les nœuds N_x et N_y dans le graphe.

Reprenons à titre d'exemple la hiérarchie illustrée dans la figure 3.15. Pour calculer l'intensité de l'attachement global entre la page p_4 et le contexte D_8, la première étape consiste à identifier les chaînes monotones reliant les deux nœuds représentant la page et le contexte considérés. Nous en identifions quatre (figure 3.17). Ensuite, la raideur de chacune de ces chaînes doit être calculée. Il s'avère que la chaîne $\widehat{ch} = p_4 \rightarrow D_4 \rightarrow D_7 \rightarrow D_8$ est la plus raide parmi les quatre. Ainsi, d'après l'équation 3.20 que nous venons de définir, l'intensité de l'attachement global de p_4 à D_8 est égale à la raideur de la chaîne $\widehat{ch}$, en l'occurrence 0.27. Bien que

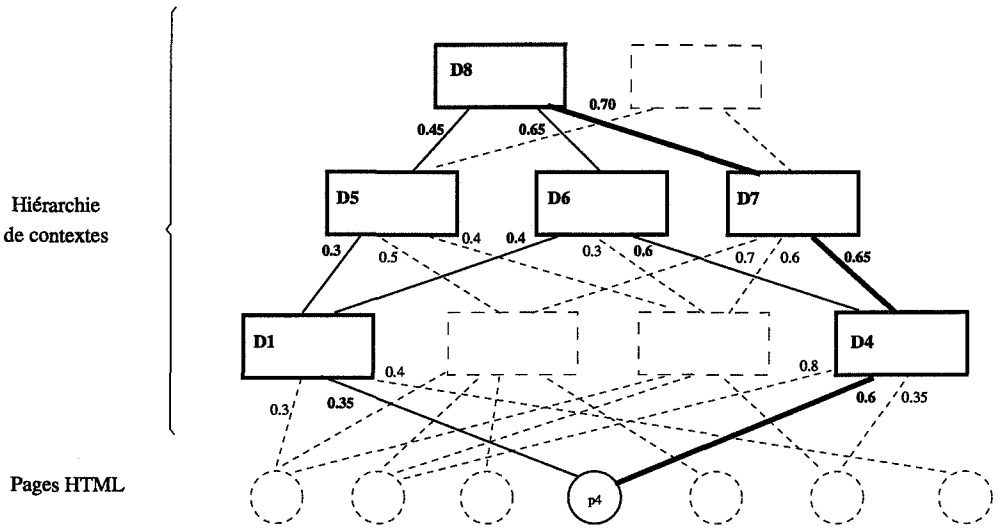

FIG. 3.17: Différentes chaînes peuvent relier une page à un contexte.

l'équation 3.20 permette de calculer l'intensité entre deux entités situées à deux niveaux non-consécutifs quelconques de la hiérarchie de contextes, dans notre contexte/application, cette formule sera toujours utilisée pour calculer l'intensité de l'attachement entre une page — plus haut nous avons supposé que les pages formaient le niveau le plus inférieur de la hiérarchie de contextes — et un contexte.

Maintenant que nous avons défini la façon de calculer l'attachement d'une page à un contexte situé à un niveau supérieur au premier niveau de la hiérarchie nous pouvons conclure la description de la troisième étape de la fonction de correspondance. Comme nous l'avons dit plus haut, la troisième étape est chargée d'estimer la pertinence finale des pages sélectionnées dans la première étape en se basant sur l'intensité de l'attachement de ces pages aux contextes sélectionnés dans la deuxième étape. Compte tenu que ni l'équation 3.10 ni l'équation 3.16 ne nous permettait de calculer l'attachement d'une page à un contexte situé à un niveau supérieur au premier, nous avons dû définir l'équation 3.20. Nous pouvons maintenant définir la pertinence finale d'une page à une requête en fonction des résultats des trois étapes de la fonction de correspondance. La pertinence finale d'une page p_x à une requête

$Q = (q_{sujet}, q_{contexte})$, $R(p_x, Q)$ est calculée d'après l'équation suivante :

$$R(p_x, Q) = Sim(p_x, q_{sujet} + q_{contexte}) \cdot (\max_{k=1,\cdots,n}(Sim(Cont_k, q_{contexte}) \cdot Att(p_x, Cont_k)) + Sim(p_x, q_{contexte})) \tag{3.21}$$

où

$$Att(p, Cont_k) = \begin{cases} \frac{1}{|Cl_k|} \cdot \sum_{p_j \in Cl_k} Compl(p_x, p_j) & si\ Niveau(Cont_k) = 1 \\ \max_{i=1,2,\cdots,m}(Att(N_x, N_y, ch_i)) & sinon \end{cases} \tag{3.22}$$

Les sémantiques des facteurs de l'équation 3.21 sont identiques à celles associées aux facteurs de l'équation 3.15 : $Cont_k$ $(k = 1, \cdots, n)$ représentent les n contextes selectionnés dans la deuxième étape de la fonction de correspondance et $Sim(A, B)$ correspond à la pertinence estimée de l'entité A à l'entité B d'après la formule du produit scalaire (équation 3.13). La seule différence entre l'équation 3.21 et l'équation 3.15 réside dans la manière dont est calculé le facteur $Att(p, Cont_k)$. Alors que dans l'équation 3.15 le facteur $Att(p, Cont_k)$ est calculé à l'aide de l'équation 3.10, dans l'équation que nous venons d'introduire, l'équation 3.10 est utilisée seulement lorsqu'il est question de calculer l'intensité d'un attachement à un contexte situé au premier niveau de la hiérarchie, i.e. un contexte C_k pour lequel $Niveau(Cont_k) = 1$. Lorsque le contexte se situe à un niveau différent du premier, i.e. $Niveau(Cont_k) > 1$, l'équation 3.20 est utilisée.

Notons que pendant l'exécution de la fonction de correspondance, les intensités des attachements entre les pages et les contextes situés au premier niveau de la hiérarchie de contextes ont déjà été calculées et se trouvent stockées dans le composant de l'index qui relie le niveau inférieur de l'index (celui associé au contenu des pages) et le niveau supérieur (celui associé au contenu des contextes) (cf. figure 3.15). Seules les intensités des attachements des pages à des contextes situés à des niveaux supérieurs au premier niveau de la hiérarchie doivent être calculées pendant le traitement de la requête.

3.6 Complexité

Dans cette section nous abordons la question de la complexité des méthodes que nous proposons pour la découverte de pages complémentaires et l'applicabilité de ces méthodes à des espaces de recherches ayant une dimension bien supérieure à celles de sites que nous avons testés dans les expérimentations décrites dans le prochain chapitre.

Analysons d'abord la complexité en termes de temps de l'algorithme associé à la première méthode — la complexité de l'algorithme associé à la deuxième méthode étant très simple à calculer une fois la complexité de la première méthode calculée.

Si nous considérons la première méthode utilisé dans le contexte d'un moteur de recherche, calculer sa complexité correspond à calculer la complexité associée à l'algorithme d'indexation. Nous avons vu dans la section 3.3.1 que l'index généré est composé de deux niveaux :

un niveau inférieur concernant les pages et un niveau supérieur concernant les contextes. La construction du niveau inférieur est exécuté en temps $O(n)$ où n est le nombre de nœuds du graphe.

Cependant la construction du niveau supérieur de l'index est beaucoup plus gourmande en termes de temps. L'algorithme associé à cette construction est composé de trois étapes : (1) le calcul de complémentarité entre toutes les pages ; (2) la clusterisation des pages selon les valeurs calculées dans la première étape et (3) l'application de la fonction d'abstraction. L'étape (2) s'exécute au maximum en temps $O(n^2)$ (où n est le nombre de pages) selon l'algorithme choisi et l'étape (3) s'exécute en temps linéaire par rapport au nombre de pages. L'étape (1) est divisée en deux sous-étapes : (1a) le calcul de la similarité entre toutes les paires de pages par rapport à leur contenu et (1b) le calcul de la proximité structurelle entre toutes les paires de nœuds du graphe sous-jacent au système hypertexte associé au site analysé.

L'étape (1a) est exécutée en temps $O(n^2)$ où n correspond au nombre de pages. L'étape (1b) constitue le vrai goulot d'étranglement de l'algorithme. Cette étape exécute des analyses sur le graphe sous-jacent aux pages à indexer. Afin de trouver le plus-court chemin entre toutes les paires de pages nous effectuons n parcours en largeur, chaque parcours partant d'un nœud différent. Ce premier calcul est associé donc à une complexité de $O(n^3)$ vu qu'un parcours en largeur s'exécute en $O(n^2)$. Cependant, les calculs des facteurs S_{ij}^{anc} (équation 3.4) et S_{ij}^{des} (équation 3.5) s'avèrent encore beaucoup plus coûteux. Il est important de souligner que les n parcours en largeur effectués lors du calcul du facteur S_{ij}^{spl} de l'équation 3.7 ne sont pas réutilisables dans le calcul de S_{ij}^{des}. La raison de cela est illustrée dans l'exemple suivant. Si le nœud x est un descendant en commun des nœuds i et j, dans l'équation 3.5, il faudra calculer le plus court chemin entre le nœud x et chacun des deux nœuds i et j avec la contrainte que le plus court chemin menant de x à i *(resp. j)* ne doit pas passer par le nœud j *(resp. i)*. La justification de cette contrainte est donnée dans la section 3.2.1.

Analysons le temps de calcul des facteurs S_{ij}^{anc} pour tous les couples (i,j) — l'analyse étant similaire pour ce qui concerne les facteurs S_{ij}^{des}. Chaque nœud du graphe peut être un ancêtre en commun de deux nœuds quelconques. Considérons comme ancêtre potentiel le nœud x. On doit calculer les plus courts chemins menant de x à tous les autres nœuds du graphe. Cependant, vu la contrainte évoquée plus haut, on va calculer les plus courts chemins de x vers tous les autres nœuds sans passer par un deuxième nœud que nous notons y. Le calcul des S_{ij}^{anc} pour tous les couples (i,j) est résumé dans l'algorithme ci-dessous :

- `Soit :` G `= (`S`,`A`)` où $n = |S|$
- `for` x `in` S (I)
  - `for` y `in` S (II)
    - $G' \leftarrow G - \{y\}$ (III)
    - `Calculer les plus courts chemins de` x `vers tous les nœuds de` G' (IV)

Vu que chacune des deux boucles imbriquées (lignes (I) et (II)) itère sur n éléments, n^2 parcours en largeur sont effectués. Étant donné qu'un parcours en largeur a une complexité de $O(n^2)$ où n est le nombre de nœuds du graphe, le calcul de S_{ij}^{anc} pour tous les couples (i,j) s'exécute en temps $O(n^4)$.

Les calculs de S_{ij}^{anc} 3.4 et S_{ij}^{des} 3.5 constituent donc la partie coûteuse de l'algorithme ce qui fait que sa complexité finale en termes de temps d'exécution est de $O(n^4)$ où n représente le nombre de pages du site analysé.

Pour ce qui concerne la deuxième méthode de découverte de pages complémentaires, la différence avec la première méthode est qu'au lieu d'un seul niveau de contextes pour les pages, plusieurs niveaux existent, l'ensemble de ces niveaux constituant la hiérarchie de contextes. D'une manière globale, cela implique que l'algorithme associé à la première méthode sera exécuté autant de fois qu'il y a des niveaux dans la hiérarchie.

L'algorithme de construction de la hiérarchie se termine lorsqu'un niveau avec un seul nœud est généré ou un nombre maximal d'itérations prédéfini est atteint. Calculons la complexité dans le cas où l'exécution de l'algorithme se termine lorsqu'un niveau avec un seul nœud est généré. Une des caractéristiques de la hiérarchie de contextes est qu'un niveau donné contient la moitié du nombre de nœuds du niveau immédiatement inférieur. Cette caractéristique, nous permet de conclure que pour un site contenant n pages, notre algorithme génère une hiérarchie composée de $\log_2 n$ niveaux où n représente le nombre de pages du site analysé. Comme le temps nécessaire pour calculer un niveau de la hiérarchie est de $O(n^4)$, la construction de la hiérarchie entière à une complexité de $O(n^4 \log n)$. Ainsi, la complexité de la deuxième méthode est de $O(n^4 \log n)$ où n représente le nombre de pages du site analysé.

3.7 Synthèse

Dans ce chapitre nous avons décrit le modèle de système de recherche d'information que nous proposons pour un système hypertexte représenté par un site Web. L'innovation consiste dans l'utilisation de l'information contextuelle des pages dans la construction de leur index.

Nous avons proposé deux méthodes pour découvrir l'information contextuelle des pages. Dans l'une des méthodes un seul niveau de contextes est découvert pour l'ensemble de pages analysés. Dans l'autre méthode plusieurs niveaux de contextes sont découverts, les contextes s'organisant dans une hiérarchie. La deuxième méthode est en fait une extension de la première. La découverte des contextes est appuyée par la découverte d'unités logiques d'information. Une unité logique d'information est le terme que nous avons choisi pour désigner un document hypertextuel dans la mesure où il s'agit d'un volume d'information auto-explicative. Finalement, la découverte des unités logiques d'information est basée sur une mesure qui estime l'intensite de la complémentarité entre deux pages appartenant à un même site. Cette mesure est une fonction du contenu des pages et de la structure qui les relie.

Nous avons également décrit comment intégrer l'information contextuelle des pages dans le fonctionnement d'un moteur de recherche. Dans ce contexte nous avons aussi suggéré un langage de requête à deux niveaux ou l'utilisateur peut distinguer dans sa requête les termes qui sont plutôt contextuels des termes qui sont centraux à son besoin d'information.

Dans le prochain chapitre nous décrivons les expérimentations que nous avons menées avec le modèle de moteur de recherche que nous avons décrit dans ce chapitre. Parmi les questions

auxquelles nous comptons pouvoir répondre avec les résultats des expérimentations, les deux principales sont : (i) la prise en compte de l'information contextuelle des pages améliore-t-elle l'efficacité des moteurs de recherche ? ; (ii) en termes d'efficacité du moteur de recherche, dans quelle mesure est-il plus intéressant de disposer d'une information contextuelle plus complète comme celle associée à la hiérarchie de contextes au lieu d'une information contextuelle condensée sur un seul niveau ?

Chapitre 4

Application : les sites de la collection WT10g

Dans ce chapitre nous abordons la partie expérimentale de notre étude. Nous commençons par décrire d'une manière générale la collection de tests que nous avons utilisée dans nos expérimentations. Ensuite nous expliquons la façon dont nous avons exploité les données composant la collection pour les adapter à nos tests. Après nous détaillons les tests effectués. Les tests se divisent en deux étapes. Dans la première nous faisons varier la valeur de certains paramètres de sorte à estimer leur valeur optimale. Une fois les valeurs des paramètres fixées, nous passons à la seconde étape où un nombre plus important de sites est utilisé pour estimer l'efficacité des méthodes que nous avons proposées pour la découverte de l'information contextuelle des pages. Nous concluons avec une analyse des résultats.

4.1 La collection WT10g

Une collection de tests dans le cadre de la recherche d'information consiste traditionnellement dans (i) un ensemble de documents ; (ii) un ensemble de besoins d'information et (iii) un ensemble de jugements. Un jugement indique si un document donné est pertinent ou non à un besoin d'information donné. Éventuellement, un jugement peut aussi spécifier un degré de pertinence[1].

Dans le cadre de nos expérimentations nous avons choisi comme collection de tests la collection WT10g issue du corpus de la conférence TREC-9 [VH00] ayant eu lieu en 2000. Cette collection a été utilisée dans le *track* Web de cette conférence [HVCB00]. Nous l'avons choisie pour deux raisons. La première est la notoriété des collections issues de TREC et par conséquent, leur statut de collections *standard* dans le domaine de la recherche d'information. La deuxième est liée à certaines caractéristiques de la collection qui convenaient à nos expérimentations.

Dans [BCH01], le processus de construction de la collection WT10g est décrit. D'une façon générale la collection a été conçue dans les objectifs de (i) modéliser une recherche réelle dans le Web et (ii) de permettre une évaluation fiable des méthodes de recherche d'information

[1] Par exemple, un document peut être *pertinent* ou *très pertinent* à une requête.

basées sur l'analyse de liens. Pour ce qui concerne le premier objectif, la collection doit être constituée : (i) d'un nombre suffisamment large et représentatif des données du Web ; (ii) d'un ensemble représentatif des requêtes posées par les utilisateurs du Web et (iii) d'un nombre de jugements suffisamment large pour que l'évaluation d'un système utilisant la collection de tests soit fiable. Pour ce qui concerne le deuxième objectif, celui associé à l'évaluation des méthodes basées sur les liens, il s'agissait de proposer une collection avec un nombre suffisamment large de liens *inter-sites* afin que méthodes basées sur les liens puissent être évaluées dans une configuration représentative de la réalité[2].

La collection est composée de 1692096 documents (pages Web) totalisant environ 11 giga-octets de données. D'après [BCH01], pour ce qui est de la structure du graphe sous-jacent à ces données, il existe 1532012 pages avec des liens entrants et 1295841 pages avec des liens sortants. Ces nombres ne sont pas exacts. En utilisant le robot que nous avons implémenté, nous avons trouvé des liens entre deux pages de la collection qui ne sont pas cités dans les fichiers livrés avec la collection et qui sont censées lister tous les liens existant entre les pages de la collection. Cela est probablement dû au fait que le robot utilisé lors de la construction de la collection pour générer les fichiers de connectivité des pages n'a pas su traiter certaines erreurs de syntaxe dans l'utilisation de la balise `<a>`.

Pour ce qui est des besoins d'information, nous avons choisi d'utiliser les cinquante besoins utilisés dans le Web *track* de TREC-9, nous les noterons $t451, t452, \cdots, t500$[3]. Dans le contexte de TREC, un besoin d'information est appelé *un topique*. Chaque topique est composé de trois champs : `<title>`, `<narrative>` et `<description>` (l'annexe B présente un exemple de topique). Comme nous l'avons dit plus haut, l'un des objectifs de la collection WT10g est de proposer un ensemble représentatif des requêtes posées par les utilisateurs du Web. Ainsi, les contenus des champs `<title>` des 50 topiques de la collection correspondent à des véritables requêtes d'utilisateurs du Web envoyées au moteur de recherche eXcite (`www.excite.com`). La longueur moyenne des requêtes est de 2.3 termes après la suppression de mots vides (la liste de mots vides que nous avons utilisée est donnée dans l'annexe C). Le contenu des autres champs des topiques, `<narrative>` et `<description>`, ont été imaginés à partir du(des) terme(s) composant le champ `<title>` par les juges assesseurs de NIST[4]. Puisque les requêtes sont très courtes et les termes utilisés sont souvent polysémiques, il est difficile de déduire le réel besoin d'information de l'utilisateur à travers sa requête. Ainsi, les contenus des champs `<narrative>` et `<description>` ne correspondent pas forcément au besoin d'information étant à l'origine de la requête de l'utilisateur.

Pour ce qui est de l'ensemble de jugements ils ont été générés à travers le processus de *pooling* utilisé dans les jugements fabriqués par TREC. La manière dont ce processus estime la liste de documents pertinents à un topique donné peut être résumé de la façon suivante.

[2]La collection WT10g a été créée à la suite de la collection WT2g bien plus petite en nombre de documents. Cette dernière collection a été utilisée dans le Web track de TREC-8. Les méthodes basées sur les liens n'ayant pas démontré l'efficacité attendue, on s'est demandé si la faible performance n'était pas due au nombre réduit des liens *inter-sites*.

[3]Le fichier contenant ces topiques est disponible sur `http://trec.nist.gov/data/topics_eng/topics.451-500.gz`.

[4]*National Institue of Standards and Technology* (`www.nist.gov`), l'institut organisateur des conférences TREC.

Chaque système participant d'une expérimentation TREC renvoie aux organisateurs de cette expérimentation les k premiers documents récupérés pour un topique donné. Ensuite, les organisateurs font juger manuellement la pertinence de chaque document renvoyé au topique en question. Les documents qui n'ont été récupérés par aucun système sont jugés non-pertinents au topique. Traditionnellement, les jugements des topiques TREC sont binaires, c.-à-d. une page est pertinente ou non à un topique et donc il n'existe pas de degré de pertinence. Cependant, pour le *track* Web de TREC-9 les jugements ont été différents : une page est classée par rapport à un topique comme non-pertinente, pertinente ou très pertinente[5].

4.2 La construction et le choix des sites

Le modèle que nous avons présenté dans le chapitre 3 s'intéresse à la recherche d'information au sein d'un système hypertexte limité et non pas à un système hypertexte comme celui sous-jacent au million et demi des pages composant la collection WT10g. Dans le contexte de cette collection, nous avons considéré un système hypertexte limité comme étant celui composé par les pages d'un site.

Or, plutôt que de considérer la collection WT10g comme une collection de pages, nous pouvons la considérer comme une collection de sites. D'après [BCH01], il existe 11680 sites dans la collection. Cependant, notre analyse de la collection nous a permis d'identifier plusieurs pages qui, bien que reportées comme appartenant à des sites différents, appartenaient à un même site. Ce type de situation est dû à la possibilité de faire référence à une machine selon son nom symbolique ou selon son adresse IP. À travers l'utilisation d'une heuristique qui consistait à analyser le champ `<DOCHDR>` associé à chaque page de la collection (cf. exemple de document dans l'annexe A) nous avons pu identifier quelques correspondances entre les noms symboliques des machines et leurs adresses IP. À l'aide des correspondances découvertes par cette heuristique, nous avons identifié 11673 sites dans la collection.

Puisque WT10g peut être considérée comme étant une collection de sites, nous pouvons donc l'utiliser pour tester notre système. Une propriété importante de la collection est que les sites qui la composent sont censés être complets dans la collection, c.-à-d., s'il existe une page d'un site donné dans la collection, toutes les autres pages de ce même site doivent aussi être présentes. Cette propriété est importante pour la validité des tests de notre système puisque ce dernier dépend entre autres du graphe sous-jacent aux pages d'un site. L'absence de certaines pages dans la collection rendrait le graphe sous-jacent au site incomplet ce qui pourrait fausser le calcul de la proximité structurelle entre deux pages du site.

Comme nous l'avons précisé dans la section 3.1, par site nous entendons un ensemble de pages associées à une thématique commune et publié comme un ensemble cohérent d'information. Nous avons supposé que les sites composant la collection correspondaient à cette définition. Dans le tableau 4.1 nous proposons quelques statistiques sur la structure de la collection en termes de sites et non pas de pages.

[5]Le fichier contenant les jugements est disponible sur `http://trec.nist.gov/data/qrels_eng/qrels.trec9.main_web.gz`. Dans ce fichier la dernière colonne peut avoir les valeurs 0, 1 et 2. Ces valeurs signifient que la page est non-pertinente, pertinente et très pertinente respectivement à un topique donné. L'annexe D présente une partie de ce fichier.

TAB. 4.1: Propriétés de la collection WT10g en termes de sites

Quantité	Description
11 673	sites
145	moyenne de pages par site
150 720	liens *inter-sites*
9977	sites avec inlinks
8998	sites avec outlinks

Afin que les conclusions tirées des expérimentations décrites plus bas aient plus de légitimité il faudrait que les sites de la collection soient représentatifs de la diversité de sites existant dans le Web. Malheureusement, comme nous allons voir dans la suite, nous n'avons pas d'indications si les sites sont vraiment représentatifs ou non.

Afin de caractériser un site nous pouvons choisir des propriétés telles que son nombre de pages, la densité de liens entre ses pages, le type de thématique (commercial, scientifique, pédagogique, etc.), les degrés de stratification ou compacité du graphe sous-jacent au site (mesures proposées pour caractériser des systèmes hypertextes dans [Bot93]), etc.

La méthode utilisée dans la construction de la collection WT10g visait à choisir un sous-ensemble du Web qui maintienne certaines propriétés que l'on croit associées au Web. Cependant seule la propriété concernant la taille des sites en nombre de pages semble avoir été considérée dans la construction de la collection. Huberman et Adamic [AH00] ont suggéré que la distribution de la taille des sites dans le Web peut être modélisée par la loi $P(n_s) = Cn_s^{-\beta}$ où C et β sont des constantes. La distribution de la taille des sites dans WT10g respecte cette distribution [BCH01]. Lawrence et Giles [LG99a][LG99b] ont estimé qu'en février 1999 le Web visible était composé de 800 millions de pages éparpillées en 2.8 millions de sites ce qui fait une moyenne de 289 pages par site. Le nombre moyen des pages dans les sites de WT10g est de 145 pages.

Pour ce qui est de la structure du graphe sous-jacent **à un site** nous ne connaissons aucune étude qui essaye de caractériser les sites du Web selon ce critère. Par ailleurs, nous croyons que la structure d'un site — peut-être son nombre de pages aussi — doit présenter une corrélation avec le type de thématique du site. Par rapport à la structure du graphe sous-jacent **à un ensemble de sites**, dans la construction de WT10g on n'a apparemment pas cherché à reproduire la connectivité irrégulière caractéristique du Web proposée par certaines études parmi lesquelles nous pouvons notamment citer celle de Broder et al. [BKM+00][6]. Au contraire, on a plutôt souhaité construire une collection suffisamment dense en termes

[6] À l'instar d'autres études moins récentes telle que [Bra96], [BKM+00] associe une connectivité assez irrégulière au Web, le divisant en quatre grands ensembles de pages de tailles équivalentes. Un premier ensemble appelé *cœur* est composé de pages bien connectées entre elles et sert de pont entre deux autres ensembles, un ensemble nommé *novice* dont les pages pointent sur les pages de l'ensemble cœur, et un autre ensemble nommé *introverti*

de liens *inter-sites* afin de proposer une collection dont les propriétés seraient compatibles avec celles présupposées par les hypothèses associées à la plupart des méthodes basées sur les liens, à savoir, un Web composée de pages bien connectées à des pages d'autres sites.

Par rapport au type de thématique des sites dans le Web, en 1999 Lawrence et Giles [LG99b] ont estimé que 86% des sites étaient de type commercial et que moins de 10% des sites avaient un contenu pédagogique ou scientifique. Nous n'avons pas d'indications sur la distribution de sites dans WT10g par rapport à ce critère.

Malgré le doute sur la réelle représentativité des sites de la collection WT10g par rapport aux sites présents dans le Web, nous avons choisi d'effectuer nos expérimentations sur les sites de la collection. Des 11673 sites présents dans la collection, seuls 871 contiennent des pages pertinentes aux topiques que nous utilisons dans les expérimentations ($t451, t452, \cdots, t500$). Pour affirmer cela nous nous basons sur les jugements fournis par TREC qui mettent en relation les pages de la collection et les topiques.

Afin de tester notre modèle l'idéal serait d'utiliser des sites sélectionnés en fonction des deux critères suivants :

1. le nombre de pages pertinentes d'un site pour un topique donné;
2. le nombre de topiques auxquels un site est pertinent[7].

L'idée sous-jacente au premier critère est de choisir des sites ayant un nombre élevé de pages pertinentes à un topique donné. Si l'on teste notre modèle sur un site qui contient uniquement un nombre réduit de pages pertinentes à un topique donné, les conclusions que l'on pourra tirer sur l'efficacité du moteur de recherche intégré au site seront peut-être biaisées. La raison de cela est que la mesure d'évaluation que nous avons utilisée pour évaluer l'efficacité du système sur un site est basée sur la précision, une mesure traditionnellemment utilisée pour évaluer les systèmes de recherche d'information. Par rapport à une requête donnée, plus il y a des documents pertinents connus, plus l'analyse de la précision est révélatrice de l'efficacité du système de recherche d'information considéré.

Pour ce qui est du deuxième critère, les sites pertinents à un nombre élevé de topiques sont préférés aux sites pertinents à un nombre réduit de topiques. L'intention ici est de tenir compte des situations où pour une raison donnée le système que nous proposons soit très efficace pour trouver des pages pertinentes à un topique dans un site mais inefficace pour un deuxième topique auquel le site testé est pertinent aussi. L'analyse de ce type de situation pourrait éventuellement donner des indices sur un type particulier de requête pour lequel notre système serait plus efficace.

Cependant, comme nous allons voir dans la suite, il y a un nombre réduit de sites dans la collection WT10g qui répondent à ces deux critères. Ainsi nous avons dû relâcher les

dont les pages sont pointées par les pages de l'ensemble cœur. Le quatrième ensemble nommé *vrille* a une taille équivalente à celle des trois autres ensembles. Les pages de cette ensemble pointent uniquement soit sur des pages de l'ensemble *novice* soit sur des pages de l'ensemble *introverti*.

[7]Nous considérons qu'un site est pertinent à un topique s'il contient au moins une page qui est pertinente à ce topique.

contraintes imposées par ces critères pour sélectionner les sites à tester dans les deux étapes expérimentales que nous expliquons dans les prochaines sections.

4.3 Les tests effectués

Dans cette section nous décrivons les tests que nous avons effectués. Vu l'existence de certains paramètres dans notre système, nous avons choisi d'abord une douzaine de sites pour trouver des valeurs optimales pour les paramètres. Ensuite, une fois les valeurs des paramètres fixées, nous avons testé notre système sur un nombre significatif de sites présents dans la collection WT10g qui sont pertinents à au moins un topique de l'ensemble de topiques que nous avons utilisé. Avant de détailler notre méthodologie expérimentale nous rappelons dans la suite les paramètres utilisés par notre système.

4.3.1 Analyse et rappel des paramètres utilisés

Les paramètres utilisés par notre système sont les suivants :

1. p_1 : le paramètre α qui permet de varier l'influence des deux termes de l'équation 3.1 dans le calcul de la complémentarité d'une page ;
2. p_2 : les paramètres β, γ et λ qui définissent l'importance des différents facteurs structuraux dans le calcul de la complémentarité structurelle entre deux nœuds (cf. équation 3.7) ;
3. p_3 : la méthode de clusterisation à utiliser lors du regroupement de pages complémentaires (cf. section 3.2.2) ;
4. p_4 : la méthode utilisée pour calculer le résumé d'un nœud contextuel à partir des nœuds dont il est issu (cf. section 3.2.3).

Les valeurs possibles pour chacun des paramètres sont illustrées dans le tableau 4.2. Notons que si l'on énumère toutes les combinaisons possibles des valeurs des paramètres on arrive à 210 possibilités. Nous n'avons pas analysé toutes ces possibilités. Dans la prochaine section nous décrivons comment nous avons procédé pour fixer les valeurs de chacun de ces paramètres.

4.3.2 Méthodologie expérimentale

Nos expérimentations se sont déroulées en deux étapes. La première étape peut être considérée comme une phase d'entraînement du système sur un nombre réduit de sites. Dans la deuxième étape, le système est appliqué à un grand nombre de sites de la collection.

Première étape

L'objectif de **la première étape** a été de fixer les valeurs des paramètres listés dans le tableau 4.2. Nous avons choisi un ensemble de 12 sites et 12 requêtes pour cette première étape des expérimentations.

Pour chaque couple (*site*, *requête*) on a effectué un nombre fixe d'itérations que nous évoquerons plus bas. Une itération consiste à (i) indexer le site en utilisant une combinaison des valeurs pour les quatre paramètres du système et (ii) soumettre une requête au moteur de recherche utilisant l'index généré dans (i) et récupérer les réponses renvoyées par le moteur.

Dans cette première étape nous avons utilisé un seul niveau de la hiérarchie de contextes pour améliorer l'index des pages. Ainsi la méthode de découverte de la complémentarité des pages utilisée est celle décrite dans la section 3.2, l'intégration de cette information dans un moteur de recherche étant traitée dans la section 3.3.

Dans un premier temps nous expliquons le paramétrage des itérations qui nous permettront d'estimer les valeurs optimales pour les paramètres. Ensuite nous abordons la construction des requêtes à partir des topiques TREC utilisées dans les itérations.

Comme cité plus haut, il existe 210 combinaisons possibles pour les valeurs des quatre paramètres. Plutôt que d'essayer les 210 combinaisons nous avons opté pour un nombre de combinaisons plus petit. D'abord nous avons associé une valeur par défaut à chacun des quatre paramètres. Les valeurs par défaut ont été choisies d'une manière arbitraire, uniquement basée sur notre intuition. Les valeurs par défaut des paramètres sont imprimées en gras dans le tableau 4.2.

TAB. 4.2: Les valeurs possibles pour les quatre paramètres des méthodes de découverte des pages complémentaires

Paramètres	**Valeurs possibles**
p_1	0, 0.25, **0.5**, 0.75, 1
p_2	{1, 0, 0}, {0, 1, 0}, {0, 0, 1}, {0.5, 0.5, 0} {0.5, 0, 0.5}, {0, 0.5, 0.5}, **{0.333, 0.333, 0.333}**
p_3	*méthode de l'étoile*, ***méthode du lien complet***, *méthode de Ward*
p_4	seuil moyen, **généralisation de** $tf \times idf$

Considérons que nous voulions définir la valeur optimale du paramètre p_1. L'idée générale pour ce faire consiste à fixer les valeurs des autres paramètres, i.e. p_2, p_3 et p_4, à leur valeur par défaut et faire varier la valeur du paramètre que nous voulons analyser, en l'occurrence p_1, sur toutes les valeurs possibles qui lui sont associées.

À la fin de ce processus nous mesurons l'efficacité du moteur de recherche dans chacune des configurations possibles (pour ce qui concerne p_1 il existe 5 configurations possibles). Ainsi, pour un site donné, nous avons cinq mesures d'efficacité, une pour chaque valeur du paramètre que nous sommes en train d'analyser, en l'occurrence p_1. Ce processus est effectué pour chaque couple $(site, requête)$. À la fin de ce processus, nous avons pour **chaque valeur possible pour le paramètre** p_1 un total de n mesures d'efficacité où n est le nombre de couples $(site, requête)$. Finalement nous calculons l'efficacité finale associée à chaque valeur possible de p_1 comme étant la moyenne des n mesures d'efficacité générées par les itérations sur les n couples. La valeur associée à la moyenne la plus élevée est celle retenue comme valeur optimale pour le paramètre p_1. Le processus est identique pour les quatre autres paramètres.

Considérons $C(p_i)$ comme étant le nombre de valeurs possibles pour le paramètre p_i. L'analyse des valeurs optimales au sein d'un site génère $(\sum_{i=1}^{n} C(p_i)) - (n-1)$ itérations différentes où n est le nombre de paramètres. Dans notre contexte, n vaut 5 et l'analyse de l'efficacité du système sur un site entraînera $(5+7+3+2-(4-1)) = 14$ itérations. Si nous voulions faire une analyse exhaustive, $(5 \times 7 \times 3 \times 2) = 210$ itérations seraient nécessaires.

Comme nous l'avons précisé ce processus est effectué pour chaque couple $(site, requête)$. Ainsi, pour estimer les valeurs optimales des quatre paramètres, $k \times 14$ itérations sont nécessaires — k est le nombre de couples $(site, requête)$, en l'occurrence 12.

La mesure que nous utilisons pour évaluer l'efficacité d'un moteur de recherche est la *précision moyenne*[8]. Pour une requête donnée, la précision moyenne est calculée de la façon suivante. D'abord on calcule la précision après que chaque document pertinent à la requête est récupéré par le moteur. Si un document pertinent n'est pas récupéré par le moteur la précision *associée à ce document* vaudra 0. On obtient ainsi pour chaque requête autant de valeurs de précisions que cette requête a de documents pertinents, la moyenne de toutes ces valeurs est ensuite calculée.

Nous avons appelé une itération le processus composé d'une indexation suivie de l'appariement d'une requête aux documents indexés d'où résulte une liste de documents considérés pertinents à la requête. Nous avons expliqué comment on a choisi les itérations, en termes de paramètres du système, qui nous ont permis d'estimer les valeurs optimales des paramètres. Il nous reste à préciser comment nous avons construit les requêtes utilisées dans les itérations.

Le langage de requête que nous avons proposé dans la section 3.3.2 est composé de deux champs : le sujet de la requête (q_{sujet}) et le contexte de la requête ($q_{contexte}$). Or, les topiques TREC dont nous disposons sont composées de trois champs : `<title>`, `<description>` et `<narrative>`. Pour construire les requêtes de cette première étape d'expérimentations nous avons utilisé uniquement le champs `<title>` des topiques. Nous avons associé les termes du champ `<title>` au champ q_{sujet} mais aussi au champ $q_{contexte}$ des nos requêtes.

L'hypothèse sous-jacente à l'association des termes de la requête non seulement au sujet de la requête mais aussi au contexte de la requête est que, bien que les termes de la requête traduisent — plus ou moins bien — l'information qu'une page pertinente doit véhiculer, ces termes ne se trouvent pas forcément dans la page elle-même, ils peuvent se trouver dans ses pages contextuelles. Nous pourrions concevoir une heuristique pour essayer de distinguer les termes de la requête qui seraient plutôt liés aux contextes des pages pertinentes à la requête, et les associer au champ contexte dans notre requête à deux champs. Cependant, vue la longueur moyenne des requêtes que nous utilisons, soit 2.3 termes, ces termes n'existent pas toujours.

[8]Les mesures d'évaluation les plus connues dans la recherche d'information sont *le taux de précision* et *le taux de rappel*. Si l'on considère la liste de réponses renvoyés par un système en réponse à une question, la *précision* correspond au pourcentage des documents récupérés qui sont connus comme étant pertinents. Le *taux de rappel* correspond au pourcentage de tous les documents connus comme étant pertinents qui ont été récupérés.

À travers les itérations que nous venons de décrire les valeurs optimales des quatres paramètres listés dans le tableau 4.2 sont estimées et la première étape des expérimentations s'achève. Passons donc à la description de la deuxième étape des expérimentations.

Deuxième étape

L'objectif de cette deuxième étape des expérimentations est d'évaluer *indirectement* l'efficacité des deux méthodes de découverte de pages complémentaires que nous avons proposées dans les sections 3.2 et 3.4. Nous l'évaluons indirectement puisque l'efficacité que nous calculons réellement est celle des moteurs de recherche qui intègrent l'information des pages complémentaires. Nous supposons que l'efficacité du moteur de recherche intégrant l'information contextuelle et l'efficacité de la méthode de découverte des pages complémentaires sont directement proportionnelles.

L'idée de départ consistait à tester les 871 sites de la collection qui sont pertinents à au moins l'un des 50 topiques que nous avons utilisés. Cependant, pour une question de temps de calcul (cf. section 3.6) nous avons sélectionné pour les tests les sites contenant au plus 700 pages. Avec cette limite sur le nombre de pages nous sommes passés de 871 sites à 641 sites. D'autres facteurs évoqués dans la section 4.4.2 ont fait que l'ensemble des sites testés a été encore réduit à 523 sites. Pour chaque site nous avons généré des requêtes pour tous les topiques auxquels le site est pertinent.

Dans cette deuxième étape d'expérimentations nous avons voulu comparer l'efficacité de trois configurations possibles pour un moteur de recherche :

1. un moteur de recherche disposant d'un index traditionnel plat (index construit sans l'utilisation de l'information contextuelle des pages) ;
2. un moteur de recherche disposant d'un index intégrant un seul niveau de nœuds contextuels ;
3. un moteur de recherche disposant d'un index intégrant la hiérarchie de contextes.

Bien entendu la configuration que nous avons considérée comme référence est celle où l'index des pages est plat ; l'index d'une page étant dérivé seulement de son propre contenu.

Par rapport aux requêtes que nous avons utilisées dans les expérimentations de cette deuxième étape, nous avons dérivé deux types de requête différents, que nous noterons R_{tt} et R_{tn}, à partir de chaque topique TREC. Les requêtes du type R_{tt} sont construites de la même façon que celles utilisées dans la première étape des expérimentions, c.-à-d. le contenu du champ `<title>` des topiques est associé aussi bien au *sujet de la requête* (champ q_{sujet}) qu'au *contexte de la requête* (champ $q_{contexte}$). Pour les requêtes de type R_{tn}, nous avons associé au *sujet de la requête* le contenu du champ `<title>` du topique et, pour ce qui concerne le *contexte de la requête*, nous lui avons associé le contenu du champ `<narrative>` des topiques.

Pour ce qui concerne les requêtes de type R_{tn}, dans le champ `<narrative>` des topiques nous trouvons une explication plus détaillée du besoin d'information exprimé par les mots-clés composant le champ `<title>`. L'hypothèse sous-jacente à l'association du contenu du champ `<narrative>` au champ *contexte de la requête* est que le champ `<narrative>` sert

à contextualiser le besoin d'information originalement spécifié par le(s) terme(s) contenu(s) dans le champ `<title>`.

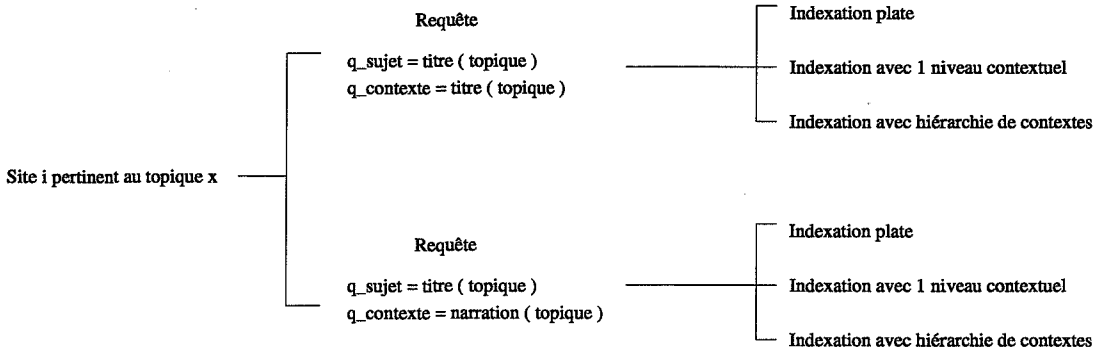

FIG. 4.1: Tests effectués pour un site i pertinent à un topique x.

Prenons comme exemple un site pertinent à plus d'un topique. Pour tester ce site nous l'indexons en créant la hiérarchie de contextes. Ensuite, pour chaque topique t_i auquel le site est pertinent nous exécutons deux itérations : une première itération où une requête de type R_{tt}, dérivée de t_i, est utilisée et une deuxième itération où une requête de type R_{tn}, dérivée aussi de t_i, est utilisée. Pour chacune des deux requêtes, le moteur de recherche que nous avons implémenté répond avec trois listes de documents qu'il a considérés comme étant pertinents aux requêtes. La première liste est issue de l'utilisation par le moteur d'une fonction de correspondance traditionnelle pour estimer le degré de pertinence des pages, i.e. la valeur de $R(p_x, Q)$. Pour cela, les index des pages qui sont utilisés sont ceux générés seulement à partir du contenu des pages et la valeur $R(p_x, Q)$ est caculée d'après la formule qui suit :

$$R(p_x, Q) = Sim(p_x, Q) = \sum_{k=1}^{\rho} w_{ik} \cdot w_{qk} \tag{4.1}$$

où Q est la requête composée de $q_{sujet} + q_{contexte}$, w_{ik} représente le poids associé au terme t_k dans la page p_x, w_{qk} représente le poids associé au terme t_k dans la requête Q et ρ est le nombre de termes différents présents dans les pages qui forment le site analysé. Bien entendu, la pondération des termes trouvés dans les pages et dans les requêtes est faite selon les équations 3.3 et 3.14 respectivement.

Le calcul de l'équation 4.1 ne reprend que le premier facteur des formules 3.15 et 3.21, respectivement adoptées par une indexation contextuelle utilisant le premier ou tous les niveaux de la hiérarchie de contextes.

Dans la deuxième liste les classements sont établis à partir des index générés à partir du contenu des pages et du premier niveau de la hiérarchie de contextes (cf. section 3.3) ; la fonction de correspondance utilisée pour calculer $R(p_x, Q)$ est l'équation 3.15. Finalement, pour ce qui concerne la troisième liste, les classements sont établis à partir des index générés

à partir du contenu des pages et de la hiérarchie de contextes complète (cf. section 3.5) ; la fonction de correspondance utilisée dans le calcul de $R(p_x, Q)$ est l'équation 3.21.

4.4 Les résultats

Dans cette section nous exposons et commentons les résultats obtenus par les deux étapes expérimentales que nous venons de décrire.

4.4.1 Résultats sur un ensemble réduit de sites : estimation des valeurs des paramètres

Pour la première étape des expérimentations nous avons sélectionné 12 couples ($site_i$, $topique_j$) — le site $site_i$ étant pertinent au topique $topique_j$ — pour trouver les valeurs optimales des quatre paramètres listés dans le tableau 4.2. Deux critères ont été utilisés pour choisir ces couples : le nombre de pages de $site_i$ et le nombre de pages de $site_i$ qui sont pertinentes au topique $topique_j$. Pour ce qui est du nombre de pages nous avons choisi des sites contenant environ 200 pages (valeur proche de la moyenne de 145 pages par site, caractéristique de la collection). Nous avons également inclus des sites ayant un nombre de pages bien supérieur et inférieur à la moyenne.

Par rapport au nombre de pages pertinentes à un topique nous avons choisi des sites ayant au moins 5 pages pertinentes pour un même topique. De cette façon nous avons pu utiliser la mesure de la précision moyenne pour évaluer l'efficacité du moteur de recherche avec les différentes combinaisons de valeurs des paramètres. Si nous analysons la distribution de pages pertinentes par topiques et par site, nous nous aperçevons que lorsqu'un site est pertinent à un topique, il contient souvent un nombre très réduit de pages qui sont pertinentes à ce topique. Le cas le plus courant étant celui où un site contient une seule page pertinente à un topique donné.

Dans le tableau 4.3 nous proposons une description sommaire des 12 sites que nous avons testés dans cette première étape des expérimentations. La quatrième colonne libellée *Max. topique* spécifie le topique pour lequel le site contient le plus de pages pertinente et la cinquième et dernière colonne indique le nombre de pages pertinentes à ce topique. Les couples que nous avons choisi d'utiliser dans cette première étape sont composés des 12 sites et des *max. topique* qui leur sont associés, par exemple, `(online96.com:80,t462)` et `(www.aclu.org:80,t464)`.

Tel que nous l'avons expliqué dans la section 4.3.2 nous avons exécuté les 14 itérations pour chacun des 12 sites listés dans le tableau 4.3. Les résultats de ces itérations sont présentés paramètre par paramètre dans les tableaux 4.4, 4.5, 4.6 et 4.7. Dans chacun des tableaux est illustrée la précision moyenne associées aux différents valeurs possibles pour le paramètre dont il est question dans le tableau. Rappelons que dans les itérations de cette première étape nous avons utilisé une indexation de pages qui bénéficie d'un seul niveau de contextes. À titre indicatif, nous avons aussi inclus dans chaque tableau la précision moyenne des itérations où l'indexation plate a été utilisée, i.e. l'indexation où seul le contenu des pages est utilisé dans la construction de leur index. Rappelons qu'une indexation plate ne dépend d'aucun des quatre paramètres qui font l'objet de cette section.

TAB. 4.3: Propriétés des sites utilisés dans la première étape des expérimentations

Machine hôte et port	**# pages**	**# topiques pertinents**	**Max. topique**	**# pages pertinentes à max. topique**
online96.com:80	256	1	t462	7
www.aclu.org:80	570	4	t464	5
www.curbet.com:80	135	1	t457	5
www.safaar.com:80	281	1	t458	7
www.charity.org:80	65	1	t453	12
www.kdux.com:80	140	1	t494	7
www.copywriter.com:80	44	1	t474	6
www.druginfonet.com:80	306	2	t454	5
www.sando.com:80	104	2	t478	7
www.vitapro.com:80	102	2	t453	5
www.eatright.org:80	252	2	t489	5
ptr.ok-connect.com:80	567	1	t493	5

La tableau 4.4 analyse le paramètre α qui pondère l'influence du contenu des pages et de la structure qui les relie dans l'estimation de la complémentarité entre elles (cf. équation 3.1). Nous avons testé cinq valeurs et celle qui a généré le meilleur résultat en termes de précision à été la valeur 0.75. C'est donc cette valeur que nous avons choisie comme valeur optimale pour le paramètre p_1. Il est cependant vrai que la précision qui lui est associée est très proche de celle générée par la situation où p_1 est positionné à 0.5 valeur que nous avions définie comme valeur par défaut. Finalement, la précision la plus élevée atteinte en faisant varier la valeur de p_1 est à peine supérieure (+2.28%) à celle associée à une indexation plate des pages.

Pour ce qui concerne le tableau 4.5 nous avons trouvé la combinaison $\{0, 0, 1\}$ comme étant la meilleure pour le paramètre p_2. Les précisions associées aux différentes valeurs possibles pour p_2 sont bien plus serrées que celles de l'analyse du paramètre p_1. Si l'on considère qu'une différence inférieur à 5% dans la précision n'est pas significative, on peut considérer que les 7 combinaisons que nous avons testées pour les paramètres β, γ et λ génèrent des précisions équivalentes. Malgré cela, nous avons choisi la combinaison $\{0, 0, 1\}$ qui a généré une précision moyenne un peu supérieure à celle générée par la combinaison $\{0.333, 0.333, 0.333\}$.

Pour ce qui est du paramètre concernant l'algorithme de clusterisation à utiliser dans le regroupement des pages complémentaires, la précision générée par le moteur lorsque l'algorithme du lien-complet est utilisé est nettement supérieure à celles générées par les deux autres algorithmes comme on le voit dans le tableau 4.6. Puisque le *lien-complet* est un algorithme qui génère des clusters disjoints, l'indexation qui nous allons utiliser dans la prochaine étape des expérimentations considérera qu'une page ne peut appartenir qu'à un seul *document*. Tel que nous l'avons évoqué dans la section 2.2.1 il s'agit d'une simplification. Cependant, d'après nos premières expérimentations, l'algorithme de l'étoile semble ne pas être efficace dans le regroupement des pages complémentaires. Une extension de notre étude

TAB. 4.4: Valeurs concernant l'estimation de la valeur optimale du paramètre α de l'équation 3.1

Paramètre p_1	
valeurs possibles	précision moyenne
0	0.3944
0.25	0.4288
0.5 (défaut)	0.4482
0.75	0.4526
1	0.4402
précision moyenne pour une indexation plate	0.4425
valeur optimale pour p_1	0.75

TAB. 4.5: Valeurs concernant l'estimation de la valeur optimale des paramètres β, γ et λ de l'équation 3.7

Paramètre p_2	
valeurs possibles	précision moyenne
$\{1, 0, 0\}$	0.44011
$\{0, 1, 0\}$	0.44115
$\{0, 0, 1\}$	0.44822
$\{0.5, 0.5, 0\}$	0.43303
$\{0.5, 0, 0.5\}$	0.44006
$\{0, 0.5, 0.5\}$	0.44292
$\{0.333, 0.333, 0.333\}$ (défaut)	0.44820
précision moyenne pour une indexation plate	0.4425
valeur optimale pour p_2	$\{0, 0, 1\}$

pourrait inclure l'analyse du comportement d'autres algorithmes de clusterisation produisant des clusters non-disjoints.

Par rapport au paramètre p_4, nos analyses (cf. tableau 4.7) indiquent que la méthode du seuil moyen est meilleure que la méthode de la généralisation de $tf \times idf$ dans le calcul du contenu d'un nœud contextuel. Alors que cette dernière méthode génère une précision très proche de celle générée par l'indexation plate, la méthode du seuil moyen, plus simple, affiche un gain de 6.3% par rapport à l'indexation plate.

Ainsi les valeurs optimales déduites à partir de ces premiers tests pour les paramètres listés dans le tableau 4.2 sont les suivantes : (i) la valeur α pour 0.75; (ii) les valeurs 0, 0 et 1 pour β, γ et λ respectivement ; (iii) la méthode de clusterisation du lien-complet et (iv) la méthode du seuil moyen.

TAB. 4.6: Valeurs concernant l'estimation de la valeur optimale du paramètre qui définit la méthode de clusterisation

Paramètre p_3	
valeurs possibles	précision moyenne
méthode de l'étoile	0.3423
méthode du lien complet (défaut)	0.4482
méthode de Ward	0.3681
précision moyenne pour une indexation plate	0.4425
valeur optimale pour p_3	*méthode du lien complet*

TAB. 4.7: Valeurs concernant l'estimation de la valeur optimale du paramètre qui définit la méthode du calcul du contenu d'un nœud contextuel

Paramètre p_4	
valeurs possibles	précision moyenne
méthode du seuil moyen	0.4766
méthode de la généralisation de $tf \times idf$ (défaut)	0.4482
précision moyenne pour une indexation plate	0.4425
valeur optimale pour p_4	*seuil moyen*

Si l'on compare la meilleure précision moyenne des 14 combinaisons de valeurs de paramètres que nous avons testées (0.4766) avec la précision générée par l'indexation plate (0.4425) nous constatons un gain de 7.7%. Cette meilleure précision est obtenue lorsque le paramètre p_4 a comme valeur la méthode du seuil moyen et tous les autres paramètres sont positionnés à leur valeur par défaut.

Notons que par rapport aux valeurs par défaut que nous avons définies en amont de cette première étape des expérimentations, trois paramètres (p_1, p_2 et p_4) parmi les quatre existants ont des valeurs optimales différentes de leurs valeurs par défaut. Toutefois, l'analyse des tableaux concernant p_1 et p_2 montre que les précisions générées par leurs valeurs optimales sont très proches de celles générées par les valeurs par défaut. La seule vrai *surprise* est liée à la valeur optimale du paramètre p_4 qui génère une précision qui est en moyenne 6.3% supérieure à celle générée par la valeur par défaut.

Dans cette première phase d'expérimentations nous avons procédé à d'autres tests utilisant le même ensemble de sites décrits dans le tableau 4.3. Après avoir estimé les valeurs optimales des quatre paramètres, nous avons voulu comparer la performance moyenne d'une indexation qui utilise un seul niveau de la hiérarchie avec celle d'une indexation plate. Les requêtes que nous avons utilisées sont les mêmes que celles utilisées dans les expérimentations qui nous ont permis d'estimer les valeurs optimales des paramètres.

Le tableau 4.8 illustre les précisions moyennes pour chacun des 12 sites testés. Ces données nous permettent d'analyser les résultats en fonction du nombre des sites pour lesquels les approches qui utilisent l'information contextuelle des pages affichent une précision supérieure, inférieure ou égale à la précision affichée par une approche où une indexation plate est utilisée. Si nous considérons qu'une différence inférieure ou égale à 5% n'est pas significative, les données de le tableau 4.8 nous montrent que pour 7 sites une indexation qui utilise un seul niveau de contextes génère une précision supérieure à une indexation plate. Pour 2 sites les deux types d'indexation (contextuelle et plate) génèrent des précisions équivalentes. Par conséquent, l'indexation plate est plus efficace qu'une indexation qui utilise de l'information contextuelle seulement pour trois sites parmi les 12 testés.

TAB. 4.8: Taux de précision moyenne par site pour les sites utilisés dans la première étape des expérimentations

	$Prec_{ind\ plate}$	$Prec_{1\ niveau}$	**Diff. % entre** $Prec_{1\ niveau}$ **et** $Prec_{ind\ plate}$
online96.com:80	0.3030	0.2919	-3.80% (=)
www.aclu.org:80	0.5456	0.5836	**+6.96%**
www.curbet.com:80	0.9429	1.0000	**+6.05%**
www.safaar.com:80	0.2828	0.5620	**+98.72%**
www.charity.org:80	0.5963	0.6983	**+17.10%**
www.kdux.com:80	0.2428	0.2637	**+8.61%**
www.copywriter.com:80	0.2674	0.1381	-93.63%
www.druginfonet.com:80	0.4497	0.5699	**+26.73%**
www.sando.com:80	0.1519	0.1438	-5.63%
www.vitapro.com:80	0.6433	0.6433	0% (=)
www.eatright.org:80	0.1658	0.2070	**+24.85%**
ptr.ok-connect.com:80	0.9029	0.7544	-19.68%

Vu le nombre réduit de sites utilisés dans cette première étape des expérimentations, il est probable que les valeurs que nous avons estimées comme étant optimales pour les quatre paramètres (cf. tableau 4.2) ne le soient pas vraiment. D'autre part, les valeurs par défaut que nous avons prédéfinies pour chacun des paramètres peuvent influer beaucoup le calcul des valeurs optimales. Par ailleurs, ces sites peuvent présenter des caractéristiques qui ne se retrouvent pas dans un ensemble plus important de sites. Ainsi, la supériorité de l'efficacité générée par une indexation contextuelle par rapport à l'efficacité entraînée par l'indexation plate que nous avons observée pour ce premier sous-ensemble réduit de sites n'implique pas nécessairement que le même scénario se reproduira dans le contexte d'un ensemble de sites bien plus grand. Cependant, nous pensons que ces résultats donnent déjà des indications sur l'utilité de l'approche qui consiste à utiliser de l'information contextuelle dans l'index des pages.

4.4.2 Résultats sur l'ensemble de sites

Dans cette section nous exposons les résultats de la deuxième étape des expérimentations. Comme nous l'avons dit plus haut, dans cette deuxième étape nous avons fixé les valeurs des paramètres du système (cf. tableau 4.2) aux valeurs optimales découvertes dans la première étape et avons fait tourner le système avec un nombre plus important de couples ($site_i$, $topique_j$). Rappelons que seulement 12 couples ont été utilisés dans la première étape.

Nous avons souhaité tester le système sur des sites (i) étant pertinents à au moins l'un des topiques de l'ensemble que nous avons utilisé (`t451,...,t500`) et (ii) contenant au plus 700 pages. Parmi les 11673 sites de la collection WT10g seuls 871 sont pertinents à au moins un topique. Finalement, de ces 871 sites, 641 contiennent un nombre de pages inférieur ou égal à 700.

En outre, les topiques `t456`, `t481` et `t495` n'ont pas été utilisés dans les tests à cause de l'existence de chiffres dans certains de leurs 3 champs : `<title>`, `<narrative>` et `<description>`. Par exemple, le champ `<title>` du topique `t456` contient le texte « *Where can I find information on the decade of the 1920's?* » . Étant donné que notre analyseur syntaxique (parseur) supprime les chiffres du texte analysé, le texte que nous venons de citer est réduit au texte « *information decade* » après analyse syntaxique et suppression des mots vides. Dans cet exemple, le terme 1920 est fondamental dans le besoin d'information exprimé par le topique et de ce fait, son exclusion lors d'une étape de pré-traitement des requêtes nuit à l'efficacité du moteur de recherche. Pour les deux autres topiques cités la situation est similaire. De ce fait, les sites qui sont pertinents à un seul topique figurant parmi les trois topiques cités plus haut n'ont pas été utilisés dans les tests ; 118 sites ont été éliminés pour cette raison[9]. Notons que si ces topiques n'avaient pas été ignorés dans les tests, non seulement l'indexation contextuelle que nous proposons mais aussi l'indexation plate traditionnelle entraîneraient toutes les deux une baisse dans l'efficacité du système. Comme nous voulons comparer l'efficacité des deux approches, l'inclusion ou non de ces 3 topiques n'a pas une vraie influence sur les résultats des comparaisons.

Ainsi, dans cette deuxième étape des expérimentations, nous avons étudié la performance du système que nous avons proposé sur 523 sites différents. La liste des sites testés ainsi que les topiques auxquels ils sont pertinents est disponible dans l'annexe E. Pour un site donné nous avons testé tous les topiques auxquels ce site est pertinent[10]. Le nombre de couples ($site_i$, $topique_j$) testés dans cette étape s'élève donc à 620. Cela fait une moyenne de 1.2 topique pertinent par site. En analysant la liste de sites dans les annexes nous pouvons effectivement constater le nombre élevé de sites qui sont pertinents à un seul topique. Ensuite, pour chaque couple ($site_i, topique_j$) les six itérations illustrées dans la figure 4.1 (p. 128) ont été exécutées.

Après, pour les requêtes de type R_{tt} — requêtes (titre,titre) — ainsi que pour les requêtes de type R_{tn} — requêtes (titre,narration) — nous calculons la précision moyenne (sur l'ensemble de topiques auquel un site est pertinent) générée (i) par l'indexation plate, (ii) par

[9] À lui seul, le topique `t495` a entraîné l'exclusion de 98 sites des tests, i.e. dans la collection de départ 98 sites sont pertinents uniquement au topique `t495`.

[10] Rappelons qu'un site est pertinent à un topique s'il contient au moins une page qui est connue comme étant pertinente au topique.

l'indexation qui utilise un niveau de la hiérarchie de contextes et (iii) par l'indexation qui utilise la hiérarchie de contextes entière.

Comme on avait fait dans l'analyse des résultats de la première étape nous proposons dans les tableaux 4.9 et 4.10 une analyse des résultats en fonction du nombre des sites pour lesquels les approches qui utilisent l'information contextuelle des pages affichent une efficacité supérieure, inférieure ou équivalente (seules les différences supérieures à 5% par rapport à la précision entraînée par l'indexation plate sont significatives) à l'efficacité affichée par l'approche où l'indexation plate est utilisée.

TAB. 4.9: Analyse des résultats par rapport au nombre de sites où une indexation contextuelle génère une précision supérieure, inférieure ou équivalente à celle générée par une indexation plate. Requêtes de type $(titre, titre)$

Requêtes de type $(titre, titre)$		
	$Prec_{1\ niveau}$ **x** $Prec_{ind\ plate}$	$Prec_{hiérarchie}$ **x** $Prec_{ind\ plate}$
$>$	75	77
$=$	331	328
$<$	117	118

Les tableaux 4.9 et 4.10 montrent que malgré le fait que l'indexation contextuelle soit préférable à l'indexation plate pour un nombre significatif des sites testés, la performance entraînée par l'indexation contextuelle est relativement bien inférieure à celle observée dans la première étape où seuls 12 sites avaient été testés. Dans le tableau 4.8 qui résume les résultats de la première étape, nous avons montré que pour les requêtes R_{tt}, nous pouvons observer que l'efficacité entraînée par l'indexation contextuelle a été nettement supérieure à l'indexation plate pour 7 des sites testés; et, au contraire, inférieure pour seulement 3 des sites testés. Les résultats de la deuxième étape montrent une tendance contraire. Par exemple, le tableau 4.9 montre que pour les requêtes R_{tt}, une indexation contextuelle entraîne une efficacité supérieure à celle d'une indexation plate pour 75 sites (resp. 77 sites) et, au contraire, inférieure pour 117 sites (resp. 118 sites) lorsque le premier niveau (resp. tous les niveaux) de la hiérarchie de contextes est(sont) utilisé(s). La supériorité de l'indexation plate sur l'indexation contextuelle est encore plus marqué lorsque les requêtes de type R_{tn} sont utilisées (cf. tableau 4.10).

Étant donné ces résultats très défavorables nous nous sommes tournés vers une analyse de la formule de pertinence — que nous rappelons ci-dessous — utilisée par le système.

$$R(p_x, Q) = Sim(p_x, q_{sujet} + q_{contexte}) \cdot (\max_{k=1,\cdots,n} (Sim(Cont_k, q_{contexte}) \cdot Att(p_x, Cont_k)) + Sim(p_x, q_{contexte})) \quad (4.2)$$

TAB. 4.10: Analyse des résultats par rapport au nombre de sites ou une indexation contextuelle génère une précision supérieure, inférieure ou équivalente à celle générée par une indexation plate. Requêtes de type $(titre, narration)$

Requêtes de type $(titre, narration)$		
	$Prec_{1\ niveau}$ **x** $Prec_{ind\ plate}$	$Prec_{hiérarchie}$ **x** $Prec_{ind\ plate}$
$>$	72	68
$=$	261	265
$<$	190	190

Nous avons décidé de manipuler l'équation à deux endroits différents. D'abord nous avons souhaité tester le remplacement du facteur $Sim(p_x, q_{sujet} + q_{contexte})$ par $Sim(p_x, q_{sujet})$. La raison initiale de l'utilisation de $q_{contexte}$ dans $Sim(p_x, q_{sujet} + q_{contexte})$ consistait à prévoir des situations où une page constituerait un document à part entière et, de ce fait, pourrait contenir non seulement les termes associés au champ q_{sujet} de la requête mais aussi les termes associés au champ $q_{contexte}$. Or, le terme $Sim(p_x, q_{contexte})$ prévoit déjà dans une certaine mesure cette situation. En outre, le facteur $Sim(p_x, q_{sujet} + q_{contexte})$ risque de récupérer des pages qui bien qu'associées aux termes de $q_{contexte}$ peuvent ne pas être associées du tout aux termes du champ q_{sujet} et, de ce fait, le système risque d'estimer des degrés de pertinence élevés pour des pages *très génériques.*

Le deuxième endroit que nous avons décidé de manipuler dans la formule de pertinence originale a été l'intensité de l'attachement d'une page à un contexte. Au lieu de considérer que lorsqu'une page est attachée à un contexte elle l'est à un certain degré qui varie entre 0 et 1, nous avons décidé de tester une intensité d'attachement binaire. Autrement dit, plutôt que de faire varier la valeur $Att(p, Cont_q)$ (cf. équation 4.2) dans l'intervalle $[0, 1]$ nous avons décidé de tester le système avec la valeur $Att(p, Cont_q)$ pouvant prendre seulement deux valeurs, 0 ou 1. Plus précisément, si une page est attachée à un contexte, i.e. s'il existe un chemin entre la page et le contexte en question (cf. figure 3.17) — l'intensité de cet attachement vaut 1 sinon elle vaut 0. Nous avons décidé de tester cette modification puisque nous craignons que le facteur $Att(p_x, Cont_k)$ soit en train d'annuler l'influence du facteur $Sim(Cont_k, q_{contexte})$ dans l'estimation de pertinence des pages. En positionnant à 1 la valeur de $Att(p_x, Cont_k)$ lorsque la page p_x est attachée au contexte $Cont_k$ nous garantissons que l'influence de $Sim(Cont_k, q_{contexte})$ est maintenue.

Ces deux variations que nous avons souhaitées tester nous ont mené à tester trois nouvelles versions de la formule de pertinence originale. Ces différentes versions sont décrites ci dessous :

$$R(p_x, Q) = Sim(p_x, q_{sujet} + q_{contexte}) \cdot (\max_{k=1,\cdots,n}(Sim(Cont_k, q_{contexte}) \cdot Att_{bin}(p_x, Cont_k)) + Sim(p_x, q_{contexte})) \tag{4.3}$$

$$R(p_x, Q) = Sim(p_x, q_{sujet}) \cdot (\max_{k=1,\cdots,n}(Sim(Cont_k, q_{contexte}) \cdot Att(p_x, Cont_k)) + Sim(p_x, q_{contexte})) \tag{4.4}$$

$$R(p_x, Q) = Sim(p_x, q_{sujet}) \cdot (\max_{k=1,\cdots,n} (Sim(Cont_k, q_{contexte}) \cdot Att_{bin}(p_x, Cont_k)) + Sim(p_x, q_{contexte})) \quad (4.5)$$

où $Att_{bin}(p_x, Cont_k)$ vaut 1 si la page p_x est attachée au contexte $Cont_k$ et 0 sinon. Dans la suite nous noterons F_A la formule de pertinence originale, i.e. l'équation 4.2, F_B l'équation 4.3, F_C l'équation 4.4 et F_D l'équation 4.5. Notons que dans F_B ainsi que dans F_C seule une des modifications évoquées plus haut a été introduite dans la formule de pertinence alors que dans F_D les deux modifications ont été introduites ensemble.

Ensuite nous avons procédé à trois séries de tests. Dans chaque série, le système est configuré d'une manière optimale, i.e. les valeurs des paramètres sont positionnées à leurs valeurs optimales, et le système utilise une des trois différentes versions de la formule de pertinence (F_B, F_C et F_D). Les résultats issus de la série de tests où la formule de pertinence originale F_A a été utilisée ont déjà été exposés (cf. tableaux 4.9 et 4.10). Ces résultats se sont avérés très défavorables pour ce qui est de l'indexation contextuelle que nous proposons dans cette étude.

Dans la suite nous exposons et comparons ces derniers résultats avec ceux issus des trois séries de tests où les versions modifiées de la formule de pertinence originale ont été utilisées. Nous analysons ces résultats en deux parties différentes : la première analysera les résultats associés à l'utilisation des requêtes de type R_{tt} et la deuxième partie concernera les requêtes de type R_{tn}.

4.4.2.1 Analyse des résultats issus des variations des formules de pertinence pour les requêtes de type R_{tt}

Dans le tableau 4.11 nous exposons les performances générées par les quatre formules de pertinence que nous avons introduites plus haut lorsque les requêtes R_{tt} sont utilisées. Rappelons l'interprétation des valeurs affichées dans le tableau. Les valeurs de la ligne intitulé ">" correspond au nombre de sites pour lesquels l'indexation contextuelle a généré une précision moyenne supérieure d'au moins 5% à l'indexation plate. Le premier nombre de cette ligne (deuxième colonne) concerne l'indexation contextuelle qui utilise un seul niveau de la hiérarchie de contextes et le deuxième nombre correspond à la situation où l'indexation contextuelle utilise tous les niveaux de la hiérarchie de contextes. L'interprétation des lignes intitulées "<" et "=" sont analogues. Notons que les nombres indiquant la performance de la formule F_A sont ceux que l'on a déjà exposées dans les tableaux 4.9 et 4.10.

Notons que les performances des formules F_B et F_D sont quasiment identiques. Cela s'explique par le fait que la seule différence entre les deux formules se trouve dans le premier facteur du produit dans lequel la formule de pertinence peut être résumée. Dans F_B le facteur est $Sim(p, q_{sujet} + q_{contexte})$ alors que dans F_D le facteur en question est $Sim(p, q_{sujet})$. Comme nous sommes en train d'analyser les performances pour les requêtes de type R_{tt}, les requêtes utilisées dans les deux calculs, i.e. $(q_{sujet} + q_{contexte})$ pour F_B et q_{sujet} pour F_D, sont similaires dans la mesure où elles contiennent les mêmes termes, ceux du titre du topique duquel la requête a été dérivée. La seule différence est que dans $Sim(p, q_{sujet} + q_{contexte})$ les termes de la requête apparaissent en double et, par conséquent, les poids qui leur sont associés par la formule de pondération (cf. équation 3.14) sont légèrement différents de ceux auxquels ils sont associés dans le calcul de $Sim(p, q_{sujet})$. Cette différence n'entraîne pas une

TAB. 4.11: Performances des formules F_A, F_B, F_C et F_D sur les requêtes R_{tt} en termes de **nombre de sites**

Requêtes de type $(titre, titre)$			
		$Prec_{1\ niveau}$ x $Prec_{ind\ plate}$	$Prec_{hiérarchie}$ x $Prec_{ind\ plate}$
F_A	$>$	75	77
	$<$	117	118
	$=$	331	328
F_B	$>$	**85**	**80**
	$<$	**76**	**74**
	$=$	**362**	**369**
F_C	$>$	75	77
	$<$	117	118
	$=$	331	328
F_D	$>$	**85**	**80**
	$<$	**76**	**74**
	$=$	**362**	**369**

modification dans l'ordre des valeurs de pertinence finales des pages même si la différence des poids des termes de la requête modifie les valeurs de pertinence des pages. Pour cette même raison les performances des formules F_A et F_C sont identiques.

L'analyse du tableau 4.11 indique que les performances des formules F_B et F_D sont largement supérieures à celle des formules F_A et F_C. Cela indique que pour les requêtes R_{tt} il est préférable d'utiliser une intensité d'attachement binaire entre une page et un contexte plutôt qu'une intensité variant dans l'intervalle $[0, 1]$. Autrement dit, dans la formule de pertinence il est préférable d'utiliser $Att_{bin}(p_x, Cont_k)$ à $Att(p_x, Cont_k)$. Les mauvais résultats entraînés par l'utilisation de $Att(p_x, Cont_k)$ indiquent que la formule 3.22 n'est pas adéquate pour calculer l'attachement et que d'autres formules devraient être étudiées. L'utilisation de l'attachement binaire fait en sorte que l'intensité de l'attachement d'une page à un contexte n'a pas d'influence sur le calcul du terme $\max_{k=1,\cdots,n}(Sim(Cont_k, q_{contexte}) \cdot Att_{bin}(p_x, Cont_k))$ dans la formule de pertinence.

En outre, le tableau 4.11 montre aussi qu'une indexation contextuelle qui utilise tous les niveaux de la hiérarchie de contextes est équivalente à une indexation qui utilise seulement un niveau de contextes. On peut résumer les résultats du tableau 4.11 par la constatation que l'indexation contextuelle et l'indexation plate entraînent des performances similaires lorsque les requêtes de type R_{tt} sont utilisées. Un test du signe bilatéral effectué sur les résultats issus de l'utilisation de la formule F_D a montré que la différence entre l'efficacité des deux méthodes n'est pas statistiquement significative ($p < 0.05$). L'indexation contextuelle entraîne une performance supérieure à celle entraînée par une indexation plate pour environ $15,8\%$ des sites alors que l'indexation plate entraîne une performance supérieure pour environ 14.3% des sites.

Nous avons aussi comparé les performances des différentes formules en termes du nombre de couples (*site*, *topique*) pour lesquels une indexation contextuelle entraîne une performance supérieure, inférieure ou équivalente à celle entraînée par une indexation plate. Ces performances sont indiquées dans le tableau 4.12. Comme la plupart des 521 sites testés sont pertinents à seulement un topique (cf. annexe E), les performances relatives ne sont pas très différentes de celles indiquées dans le tableau 4.11. Un test du signe bilatéral a montré que pour ce qui est des résultats de l'utilisation de la formule F_D en termes de couples la différence entre la performance entraînée par l'indexation plate et l'indexation contextuelle à un seul niveau n'est pas statistiquement significative ($p < 0.05$).

TAB. 4.12: Performances des formules F_A, F_B, F_C et F_D sur les requêtes R_{tt} en termes de **nombre de couples**

Requêtes de type (*titre*, *titre*)			
		$Prec_{1\ niveau}$ **x** $Prec_{ind\ plate}$	$Prec_{hiérarchie}$ **x** $Prec_{ind\ plate}$
F_A	$>$	84	87
	$<$	144	145
	$=$	392	388
F_B	$>$	**102**	**98**
	$<$	**93**	**91**
	$=$	**425**	**431**
F_C	$>$	84	87
	$<$	144	145
	$=$	392	388
F_D	$>$	**102**	**98**
	$<$	**93**	**91**
	$=$	**425**	**431**

Les performances entraînées par les formules F_B et F_D (cf. tableaux 4.11 et 4.12) appuient dans une certaine mesure notre hypothèse de départ selon laquelle une page ne représente pas un volume d'information auto-explicative et qu'ainsi son index ne doit pas être généré uniquement à partir de son contenu. Même si d'après nos résultats l'indexation contextuelle pour les requêtes R_{tt} est aussi efficace que l'indexation plate, le fait que pour un pourcentage important de sites (ou de couples) l'indexation contextuelle ait été supérieure à l'indexation plate nous laisse présager qu'une indexation contextuelle peut effectivement améliorer la efficacité d'un système de recherche d'information dans certaines situations.

Cela dit, le système que nous avons proposé dans le chapitre trois n'a pas été conçu pour les requêtes R_{tt} où les termes associées au contexte de la requête, $q_{contexte}$, sont les mêmes que ceux associés au sujet de la requête, q_{sujet}. Le système que nous avons proposé a été conçu pour des requêtes où les termes associés au champ $q_{contexte}$ contextualiseraient les termes

associés au champ $q_{contexte}$. Nous avons simulé ce type de requête en créant les requêtes R_{tn} à partir des topiques TREC. Dans la prochaine section, nous allons donc étudier la performance des formules F_A, F_B, F_C et F_D pour la situation où les requêtes R_{tn} sont utilisées.

4.4.2.2 Analyse des résultats issus des variations de la formule de pertinence pour les requêtes de type R_{tn}

Dans le tableau 4.13 nous exposons les performances générées par les quatre formules de pertinence, F_A, F_B, F_C et F_D que nous avons énumérées dans la section précédente, pour les requêtes R_{tn} où le sujet de la requête est associé aux termes du champ `<title>` d'un topique TREC et le contexte de la requête est associé aux termes du champ `<narrative>` de ce même topique. Bien entendu, dans les tests que nous évoquons dans cette section nous avons utilisé les 523 sites et les 620 couples (*site*, *requête*) qui leur sont associés. Une dernière remarque à faire avant de passer à l'analyse des résultats est que dans le tableau 4.13 nous avons affiché la performance des différentes formules de pertinence non seulement en termes de nombre de sites mais aussi en termes de couples (nombres entre parenthèses) pour lesquels une indexation contextuelle entraîne une efficacité supérieure, inférieure ou similaire à celle entraînée par une indexation plate. Notons aussi que les données correspondant à la performance de la formule F_A ont déjà été évoquées dans le tableau 4.10.

TAB. 4.13: Performances des formules F_A, F_B, F_C et F_D sur les requêtes R_{tn} en termes de **nombre de sites et de couples**

Requêtes de type (*titre*, *narration*)			
		$Prec_{1\ niveau}$ **x** $Prec_{ind\ plate}$	$Prec_{hiérarchie}$ **x** $Prec_{ind\ plate}$
F_A	$>$	72 (84)	68 (80)
	$<$	190 (223)	190 (223)
	$=$	261 (313)	265 (317)
F_B	$>$	79 (97)	79 (97)
	$<$	160 (185)	164 (188)
	$=$	284 (338)	280 (335)
F_C	$>$	128 (144)	127 (144)
	$<$	99 (120)	102 (124)
	$=$	296 (356)	294 (352)
F_D	$>$	**136 (163)**	**134 (160)**
	$<$	**74 (86)**	**76 (89)**
	$=$	**313 (371)**	**313 (371)**

Selon les résultats nous pouvons constater que les performances des formules F_C et F_D sont proches l'une de l'autre mais nettement supérieures aux performances générées par les formules F_A et F_B. En comparant les formules F_C et F_D aux formules F_A et F_B nous remarquons que les bonnes performances de F_C et F_D doivent être dues à l'utilisation du facteur

$Sim(p_x, q_{sujet})$ au lieu du facteur $Sim(p_x, q_{sujet} + q_{contexte})$. La crainte évoquée plus haut que l'utilisation de $Sim(p_x, q_{sujet} + q_{contexte})$ associe des degrés de pertinence élevées à des pages très génériques pouvant ne pas être associées du tout au sujet de la requête semble se confirmer.

En outre, comme nous l'avions constaté dans les tests concernant les requêtes R_{tt}, l'utilisation d'une intensité d'attachement binaire entre une page et un contexte est préférable à une intensité d'attachement qui varie dans l'intervalle $[0, 1]$ et est calculée par l'équation 3.22. En effet la performance entraînée par la formule F_D est supérieure à celle entraînée par la formule F_C. Même si les données concernant ces deux formules indiquent que toutes les deux entraînent une efficacité supérieure à celle entraînée par une indexation plate, la performance de la formule F_D est nettement supérieure à celle de la formule F_C.

Comme nous l'avions évoqué plus haut, il est probable que dans certaines situations la valeur $Att(p, Cont_k)$ soit en train d'annuler l'influence du facteur $Sim(Cont_k, q_{contexte})$ dans le produit $(Sim(Cont_k, q_{contexte}) \cdot Att(p_x, Cont_k))$. Une étude sur la distribution de la valeur de $Att(p_x, Cont_k)$ devrait être faite afin d'analyser l'adéquation de l'équation 3.22 pour le calcul de $Att(p_x, Cont_k)$. Une intensité d'attachement binaire garantit que l'influence de $Sim(Cont_k, q_{contexte})$ est maintenue sur l'estimation du degré de pertinence d'une page à une requête.

Finalement, comme on a constaté dans les résultats des tests sur les requêtes R_{tt} analysés dans la section précédente, l'utilisation de tous les niveaux de la hiérarchie de contexte entraîne une performance qui est dans la plupart des cas similaire à celle entraînée par l'utilisation du premier niveau de contextes uniquement. Dans certains cas, l'utilisation de plusieurs niveaux entraîne une performance légèrement inférieure. Cela dit, les deux types d'indexation contextuelle (utilisation d'un seul niveau ou de tous les niveaux de la hiérarchie) entraînent une efficacité qui est largement supérieure à celle entraînée par une indexation plate pour l'ensemble de sites testés. Cela est particulièrement vrai lorsque la formule de pertinence F_D est utilisée. Un test du signe unilatéral a montré que la différence entre la performance entraînée par l'utilisation de F_D dans le cas d'une indexation contextuelle et celle entraînée par une indexation plate est statistiquement significative ($p < 0.05$).

En effet, les bonnes performances des formules F_C et F_D appuient notre hypothèse de départ selon laquelle une page ne doit pas être systématiquement considérée comme un document à part entière. Par rapport au contenu d'une page HTML, il existe souvent de l'information contextuelle qui se trouve à l'extérieur de la page elle-même. En outre, dans la traduction du besoin d'information d'un utilisateur des termes de deux types différents peuvent co-exister : des termes fondamentaux dans le besoin d'information de l'utilisateur, ces termes se trouvant probablement sur des pages pertinentes à la requête, et d'autres termes qui (i) servent plutôt à contextualiser les termes du premier type et (ii) ne se trouvent pas forcément dans des pages pertinentes mais dans des pages qui contiennent de l'information qui contextualise celle véhiculée par des pages pertinentes. Le processus d'estimation de la pertinence d'une page HTML à une requête doit considérer différemment les termes de la requête selon le type auquel ils appartiennent.

Cela dit, encore faut-il que la distinction des deux types de termes soit faite. On peut imaginer (i) que l'utilisateur puisse faire cette distinction lui-même lors de la formulation de sa requête où (ii) que le système intègre un module capable de faire cette distinction à travers l'utilisation d'un outil qui posséderait une connaissance des relations pouvant exister entre des termes (p.ex. un thésaurus). Cependant, si d'une part l'utilisateur ne pas se prêtera pas nécessairement à l'effort pour la distinction (nous revenons sur cette question dans la conclusion), d'autre part, le fait que pour deux besoins d'information différents les rôles de deux mêmes termes de la requête pourraient s'inverser, l'utilisation d'un module où les relations entre les termes sont pré-définies et éventuellement figées peut être problématique. Ainsi, le problème de la distinction des deux types de termes reste à étudier, et cela d'autant plus que dans le Web la probabilité qu'il existe des termes de deux types dans les requêtes est très basse vu que les requêtes soumises aux moteurs de recherche sont souvent très courtes.

4.5 Résumé

Dans ce chapitre nous avons détaillé les expérimentations que nous avons menées pour évaluer l'utilité des méthodes de découverte de pages complémentaires dans l'amélioration de l'efficacité d'un moteur de recherche.

La collection de tests que nous avons utilisée est issue des conférences TREC des deux dernières années. Pour ce qui est des documents nous avons utilisé la collection WT10g. Par rapport aux requêtes nous avons exploité les topiques $t451, \ldots, t500$.

Nos expérimentations se divisent en deux parties. Dans la première partie nous avons mené des tests sur un nombre réduit de sites afin d'estimer les valeurs optimales pour les paramètres des méthodes que nous avons proposées (cf. tableau 4.2). Une fois les valeurs optimales estimées, nous avons procédé à la deuxième étape des expérimentations où tous les sites de la collection ayant au plus 700 pages ont été testés.

Les résultats de la première étape ont laissé présager que l'utilisation de l'information contextuelle des pages au sein d'un moteur de recherche pourrait entraîner une efficacité supérieure à celle entraînée par un moteur de recherche qui n'utiliserait pas de l'information contextuelle dans l'indexation des pages. Cependant, lorsque le système a été testé avec un nombre bien plus important de sites dans la deuxième étape, les résultats se sont inversés : l'indexation contextuelle a entraîné une efficacité bien inférieure à celle de l'indexation plate (cf. tableaux 4.9 et 4.10).

Nous avons donc procédé à des manipulations sur la formule de pertinence originale en la modifiant sur deux endroits différents. Nous avons créé trois versions différentes de la formule de pertinence originale et avons effectué des tests avec ces nouvelles versions de la formule de pertinence. Finalement, nous avons comparé les résultats avec ceux issus des tests où la formule de pertinence originale a été utilisée.

L'analyse des résultats des derniers tests a été divisée en deux parties. Dans la première partie (section 4.4.2.1) nous avons analysé les résultats issus des itérations où les requêtes de type R_{tt} ont été utilisées. Les résultats ont indiqué que deux formules — F_B et F_D —

issues des variations de la formule de pertinence originale entraînent une efficacité supérieure à celle de l'indexation plate pour un nombre significatif des sites testés (cf. tableaux 4.11 et 4.12) ; l'efficacité de l'indexation contextuelle pour la totalité des sites testés reste cependant comparable à celle de l'indexation plate. Cela dit, le système que nous avons proposé dans le troisième chapitre n'a pas été conçu pour répondre aux requêtes de type R_{tt}.

La deuxième partie de l'analyse a concerné les résultats issus des itérations où les requêtes de type R_{tn} ont été utilisées (section 4.4.2.2). Rappelons que les requêtes de type R_{tn} sont les requêtes pour lesquelles le système a été originalement conçu. Notamment, la formule F_D a entraîné une efficacité largement supérieure à celle obtenue par une fonction de pertinence associée à une indexation plate (cf. tableau 4.13). Nous avons donc trouvé de fortes indications que (i) l'information contextuelle d'une page peut se trouver à l'extérieur de la page et son utilisation dans l'index de la page peut apporter une amélioration de l'efficacité du système et (ii) dans une requête il peut avoir des termes qui ne sont pas au même niveau dans l'expression d'un besoin d'information et leur distinction dans le processus d'estimation du degré de pertinence d'une page peut également apporter une amélioration de l'efficacité du système.

Finalement les résultats ont donné des indications que l'indexation contextuelle utilisant la hiérarchie de contextes entière n'entraîne pas une efficacité supérieure à celle de l'indexation contextuelle où seul le premier niveau de contextes de la hiérarchie est utilisé (cf. tableaux 4.11 et 4.12). Cela remet en cause la construction de la hiérarchie ainsi que son utilité dans le contexte d'un moteur de recherche.

Chapitre 5

Conclusion

5.1 Problème abordé

Dans cette thèse nous nous sommes intéressés à la modélisation d'un système de recherche d'information appliqué aux systèmes hypertextes. Notre motivation est issue de l'observation que sous un format hypertextuel il n'est pas toujours facile d'identifier un document[1] vue sa fragmentation en plusieurs nœuds.

Comme un document hypertextuel est souvent fractionné en plusieurs nœuds, l'information véhiculée par un nœud N_i n'est pas dérivée uniquement de son contenu mais elle est aussi fonction du contenu d'autres nœuds faisant partie du même document que N_i. Autrement dit, souvent l'information véhiculée par un nœud ne peut pas être appréhendée à partir de la seule analyse de son contenu. Il existe souvent de l'information dans certains nœuds du voisinage d'un nœud donné qui contextualise l'information contenue dans ce dernier : il s'agit de nœuds que nous avons appelés dans ce rapport *nœuds complémentaires*. Si l'on considère le problème du côté des systèmes de recherche d'information, lorsque l'on indexe un nœud à partir de son seul contenu, on risque d'avoir un index qui ne révèle pas précisément l'information véhiculée par le nœud.

Si l'on considère le Web comme étant un système hypertexte où les nœuds représentent les pages, on peut imaginer que le contenu d'une page (i) ne compose pas un document[2] à part entière et, par conséquent, l'indexation d'une page ne devrait pas être uniquement basée sur son contenu. Les moteurs de recherche traditionnels considèrent cependant les pages comme étant des documents à part entière et construisent leur index en se basant seulement ou majoritairement sur son contenu. De ce fait, les moteurs risquent de créer des index qui ne reflètent pas bien le contenu sémantique des pages. Il existe cependant quelques moteurs (e.g. *Google*) qui utilisent dans l'indexation d'une page le texte de l'ancre des liens qui pointent sur la page. L'hypothèse considérée par ces moteurs est que le texte des ancres des liens résument bien le contenu de la page pointée. On pourrait aussi considérer les textes des ancres comme des compléments au contenu de la page pointée et non pas un résumé de ce contenu. Cependant cela implique que les pages complémentaires à une page donnée

[1] Dans cette étude nous avons considéré un *document* comme étant un volume d'information auto-explicative.
[2] Dans la mesure où elle ne contient souvent pas une information auto-explicative.

pointent directement sur les pages qu'elles complémentent. Nous croyons que l'organisation de l'information dans un système hypertexte est plus complexe que celle que nous venons de considérer et qu'ainsi l'information contextuelle d'une page ne se trouve pas systématiquement sur l'ancre des liens à travers lesquels la page est pointée, mais qu'elle peut se trouver plus *loin* dans le graphe.

5.2 Proposition

Dans le but de proposer un modèle de système de recherche d'information ou, plus simplement, de moteur de recherche, adapté aux systèmes hypertextes nous avons conçu une méthode pour retrouver l'information qui contextualise le contenu d'un nœud mais qui n'est pas toujours présente dans le nœud lui-même. Bien sûr, nous avons choisi comme terrain d'application le *World-Wide Web*. Les systèmes hypertextes pour lesquels nous avons conçu notre moteur de recherche sont donc ceux associés aux sites Web. Par site Web nous entendons un ensemble de pages associées à une thématique plus ou moins commune, maintenues par une même personne ou un même groupe de personnes, et **publié comme un ensemble cohérent d'information.**

L'objectif de la découverte de l'information contextuelle des pages composant un site est bien entendu celui d'améliorer l'index de ces pages. En améliorant les index des pages nous espérons une amélioration de l'efficience du moteur de recherche associé. Dans le but de trouver l'information contextuelle des pages nous avons commencé par proposer une méthode de découverte d'**unités logiques d'information**, ULI. Une ULI est composée d'un ensemble de pages qui se complémentent entre elles représentant ainsi une information auto-explicative. Ainsi, une ULI peut être associée à un document hypertextuel. Cependant, une page peut appartenir à plusieurs documents à la fois. De ce fait, une ULI ne contient pas nécessairement de l'information concernant un seul document : on y retrouve l'information complète et auto-descriptive concernant un document donné mais on peut aussi y trouver de l'information incomplète concernant d'autres documents associés à d'autres ULIs.

Afin d'identifier les ULIs composant un site Web nous avons d'abord défini une *mesure* qui estime la **complémentarité** entre deux pages. Cette mesure est basée (i) sur la proximité structurelle entre les pages par rapport au graphe sous-jacent au site analysé et (ii) sur la similarité entre le contenu des pages. Ensuite nous avons suggéré l'utilisation d'un algorithme de clusterisation pour **regrouper les pages qui se complémentent** dans ce que nous avons appelé unités logiques d'information. Ensuite nous avons introduit une heuristique pour retrouver l'information contextuelle des pages regroupées dans une ULI à partir de leurs contenus. Il s'agit de la **fonction d'abstraction** qui crée à partir d'une ULI un nœud contextuel. Finalement nous avons défini une mesure qui estime combien une page est attachée à un contexte, l'hypothèse considérée ici est que dans une ULI il peut y avoir des pages plus centrales que d'autres, des pages qui seraient plus importantes dans la composition d'un document que d'autres. Nous identifions ces pages à travers les degrés de complémentarité existant entre les pages regroupées dans une même ULI. Il s'agit de la **fonction d'attachement.**

Une fois que les contextes des pages sont identifiés, nous avons montré comment intégrer

cette information dans les index des pages générés par un moteur de recherche. Vu que l'idée de départ était d'utiliser l'information contextuelle dans l'index des pages, la démarche *logique* ici aurait été de combiner le contenu d'une page avec le contenu de son contexte d'une certaine façon afin de générer l'index final de la page. Cependant, nous avons opté pour une démarche différente. Nous avons souhaité garder les deux contenus (celui de la page et celui associé à son contexte) séparés et créer deux index associés à une même page, l'un associé au contexte et l'autre associée au contenu de la page.

Un tel type d'index pour une page s'explique par le **langage de requête** que nous avons proposé pour le moteur de recherche. Ce langage est composé de **deux opérateurs : l'opérateur sujet et l'opérateur contexte**. À l'opérateur sujet doivent être associés les termes *fondamentaux* d'un besoin d'information. Sauf dans des cas très particuliers, de tels termes se trouvent dans le contenu d'une page pertinente. En revanche, à l'opérateur contexte doivent être associés les termes que contextualisent les termes associés au sujet de la requête. Il s'agit des termes qui bien que liés au besoin d'information de l'utilisateur, sont moins centraux que les termes associés au sujet de la requête. Ces termes ne se trouvent pas nécessairement dans des pages pertinentes au besoin d'information de l'utilisateur mais il est probable qu'ils se trouvent dans des pages complémentaires à une page pertinente. Dans la **fonction de correspondance** finale du moteur de recherche, les termes du sujet de la requête sont comparés aux index des contenus des pages et les termes du contexte de la requête sont comparés aux index des contextes des pages. Finalement, pour le calcul du degré de pertinence final d'une page le système combine (i) le degré de pertinence de son contenu au sujet de la requête ; (ii) du degré de pertinence de son(ses) contexte(s) au contexte de la requête, et (iii) de l'intensité de l'attchement de la page à son(ses) contexte(s). Bien entendu, le cas, que nous croyons d'ailleurs être rare, où une page représente une information auto-explicative est aussi prévu dans la fonction de classement.

Nous proposons aussi dans ce rapport une deuxième méthode pour découvrir les contextes des pages. Cette méthode constitue en fait une extension de la première : au lieu de découvrir un seul niveau de contextes pour les pages, nous proposons la découverte de plusieurs niveaux. Les différents niveaux de contextes s'organisent dans une hiérarchie que nous avons appelée **la hiérarchie de contextes**. Le nœuds contextuels appartenant au niveau $k+1$ de cette hiérarchie sont générés à partir des nœuds appartenant au niveau k. La méthode de construction d'un niveau de la hiérarchie à partir du niveau immédiatement inférieur peut être considérée comme une généralisation de la première méthode où les nœuds de niveau inférieur ne représentent pas des pages HTML mais des contextes. L'idée sous-jacente à la construction de la hiérarchie est de découvrir des contextes plus généraux que ceux découverts par la première méthode et ainsi de rendre possible l'identification d'une page comme pertinente à partir d'un contexte qui, à première vue, i.e. d'après les contextes révélés par la première méthode, ne serait pas considérée comme étant le sien.

Commentaires sur l'implémentation et les tests réalisés

Nous avons intégralement implémenté les techniques que nous venons d'énumérer. En effet, nous aurions pu récupérer certains logiciels dont les codes sources sont disponibles gratuite-

ment[3] afin d'adapter certaines de ses modules, e.g. le module chargé de la création et gestions des fichiers inversés composant les index, à nos besoins. Cependant, les coûts d'adaptation et d'intégration des modules récupérés pourraient s'avérer non-négligeables et difficiles à évaluer a priori. Le choix d'implémenter nous-même les modules qui pourraient éventuellement être récupérés ailleurs s'explique aussi par la certitude que nous apprendrions davantage si nous modélisions et implémentions nous même le système. Nous avons également construit un robot afin de collecter les pages composant un site. En revanche, pour tout ce qui est innovant dans notre modèle, i.e. l'estimation de la complémentarité des pages, la découverte d'unités logiques d'information, la découverte des nœuds contextuels, la construction de la hiérarchie de contextes, le langage de requête et la fonction de classement associée, l'option de récupérer et adapter des modules d'autres systèmes n'existait bien entendu pas. Notre système a été majoritairement développé dans le langage C++. Un nombre réduit de modules ont cependant été écrits en PERL, e.g. l'analyseur syntaxique des pages HTML.

Pour les tests nous avons utilisé la collection de documents WT10g issue de TREC, les topiques $t451, t452, \cdots, t500$ utilisés dans le Web *track* de TREC-9 et les jugements de pertinence associés. Notamment, dans la deuxième partie des tests nous avons trouvé des résultats qui indiquent que l'utilisation de l'information contextuelle des pages peut mener à une amélioration de l'efficacité d'un moteur de recherche. Cela est particulièrement vrai lorsque les requêtes utilisées sont cohérentes avec le modèle de requêtes à deux niveaux que nous avons défini dans la section 3.3.2. Cela a été le cas dans les tests dont nous avons exposé les résultats dans la section 4.4.2.2 : la formule de pertinence F_D (cf. équation 4.5) a généré une efficacité largement supérieure à celle entraînée par une indexation plate où l'information contextuelle n'est pas utilisée (cf. tableau 4.13).

Cela dit, il existe une nombre important de sites pour lesquels l'indexation contextuelle s'est revelée néfaste à la performance du système ayant entraîné une efficacité inférieure à celle entraînée par une indexation plate. Cela peut être un signe que la méthode de découverte de pages complémentaires que nous avons proposée dans le chapitre trois associe dans certains cas un contexte à une page qui n'est pas le sien. Par conséquent, au lieu d'améliorer la qualité de l'index de la page, l'indexation contextuelle la rend plus mauvaise. Une extension de cette étude consisterait essayer de trouver des caractéristiques des sites pour lesquels l'indexation contextuelle a bien fonctionné et des sites pour lesquels elle n'a pas fonctionné. La confrontation des deux ensembles de caractéristiques pourrait donner des indications pour une possible évolution de l'indexation contextuelle que nous avons proposée dans cette étude.

Une dernière remarque concernant les résultats obtenus est que même si l'efficacité entraînée par une indexation contextuelle qui utilise la hiérarchie de contextes entière est toujours supérieure à l'efficience entraînée par une indexation plate, elle est en contrepartie similaire et parfois même inférieure à l'efficacité entraînée par l'indexation contextuelle où seul le premier niveau de la hiérarchie est utilisée. Ce résultat négatif indique qu'un travail d'analyse de la construction de la hiérarchie est nécessaire.

Bien que les résultats obtenus soient plutôt révélateurs d'une utilité de l'information contex-

[3]Par exemple les systèmes *SMART* (`ftp://ftp.cs.cornell.edu/pub/smart`) et *Managing Gigabytes* (`http://www.mds.rmit.edu.au/mg/`).

tuelle dans l'indexation des pages, nous tenons à souligner **l'inadaptation de la collection de tests** utilisée dans nos expérimentations. Vu que le système que nous proposons est conçu pour la recherche d'information au sein d'un site, il aurait fallu tester sur un site donné un nombre de requêtes important auxquelles le site serait pertinent afin d'obtenir une évaluation plus fiable du système. En plus il aurait fallu aussi que pour une requête donnée le site contienne un nombre important de pages pertinentes. Or, par rapport à l'ensemble de jugements que nous avons utilisé, un site donné est très souvent pertinent à une seule requête à travers une seule page. Autrement dit, l'ensemble de jugements utilisé n'est pas adapté à nos tests. D'après le système de *pooling* utilisé dans la construction des jugements, si un document pertinent de la collection n'est retourné par aucun système faisant partie des expérimentations, ce document n'est pas jugé manuellement par les assesseurs et sera ainsi considéré d'office comme non-pertinent. Dans [BCH01] on affirme que l'ensemble de 70070 jugements effectuées manuellement qui mettent en relation les 1692096 pages composant la collection et les 50 topiques est suffisamment grand pour une évaluation fiable des systèmes participant aux expérimentations. Si l'on considère (i) les 523 sites que nous avons testés[4] (parmi les 871 possibles) et (ii) les 9349 jugements (parmi les 70070) concernant leurs pages nous arrivons à une moyenne de 17.9 jugements par site ce qui risque de ne pas être suffisant pour évaluer l'efficience d'un moteur de recherche quand il est appliqué site par site.

Une autre question liée à la collection de tests concerne la construction des requêtes à deux opérateurs utilisées dans nos expérimentations à partir des topiques TREC. Tel que nous l'avons mentionné dans le chapitre 4 nous avons utilisé deux types de requêtes pour tester chaque site, les requêtes R_{tt}[5] et R_{tn}[6]. Les requêtes de type R_{tt} sont représentatives des requêtes du Web, cependant vu que les deux opérateurs, sujet et contexte, sont associés aux mêmes termes, ces requêtes ne sont pas cohérentes avec l'esprit des deux opérateurs que nous avons proposés. Inversement, alors que dans les requêtes de type R_{tn} l'association des termes aux deux opérateurs de la requête est cohérente avec leur esprit[7], ces requêtes ne sont probablement pas représentatives des requêtes que les utilisateurs du Web soumettraient s'ils utilisaient le modèle de requête que nous avons proposé[8]. Cette discussion nous permet d'enchaîner avec la prochaine section où nous citons les limitations du notre modèle.

5.3 Limitations

La limitation peut-être la plus évidente de notre modèle concerne le modèle de requête que nous avons proposé. D'une part nous imaginons que pour certains besoins d'information la formulation de la requête sous le langage que nous suggérons peut s'avérer une tâche

[4] Nous avons testé les sites de la collection WT10g ayant au plus 700 pages et qui sont pertinents à au moins un topique.

[5] Les termes contenus dans le champ `<title>` d'un topique sont associés aux deux champs, sujet et contexte, des requêtes à deux niveaux.

[6] Les termes contenus dans le champ `<title>` d'un topique sont associés au champ *sujet* et les termes contenus dans le champ `<narrative>` sont associés au champ *contexte* des requêtes à deux niveaux.

[7] Si l'on considère que le contenu du champ `<narrative>` d'un topique contextualise le contenu du champ `<title>` de ce même topique.

[8] Le contenu du champ `<narrative>` des topiques est souvent trop long. Vu la taille réduite des requêtes observées dans le Web, nous imaginons que si le champ *contexte* existait, l'utilisateur y associerait un nombre réduit des termes.

cognitive très complexe pour l'utilisateur (qu'est-ce qui est central dans mon besoin d'information ? qu'est-ce qui le contextualise ?). D'autre part, nous avons tendance à penser (vu les caractéristiques des requêtes observées dans les *logs* des quelques moteurs de recherche très utilisées du Web) que l'utilisateur ne serait pas prêt à fournir l'effort pour formuler sa requête d'après le modèle de requête que nous proposons. Cependant les expérimentations que nous avons menées avec les requêtes de type R_{tt} montrent que notre modèle pourrait s'avérer utile aussi pour les requêtes traditionnelles à un seul niveau. Rappelons que les contenus des champs `<title>` des topiques TREC sont issus de véritables requêtes soumises à un moteur de recherche grand-public du Web.

La deuxième limitation de notre système est liée à la complexité des algorithmes. Dans la section 3.6 nous avons montré que la complexité de l'algorithme d'indexation proposé dans notre système est de $O(n^4)$ en termes de temps d'exécution où n correspond au nombre de pages du site indexé. Cependant, vu le nombre moyen de pages des sites de la collection WT10g, 145 pages par site, ou cette même moyenne estimée dans le contexte du Web entier par [LG99a][LG99b], 245 pages par site, notre algorithme d'indexation a un temps d'indexation tout à fait acceptable pour un nombre important de sites dans le Web. On peut alternativement considérer que notre modèle est acceptable uniquement pour les sites qui ne dépasseraient pas un nombre donné de pages.

Vu sa complexité, notre algorithme n'est bien entendu pas applicable au Web entier composé des centaines de millions de nœuds. Cependant, si l'on considère que les pages complémentaires d'une page donnée se trouvent dans le même site que cette page, l'analyse de complémentarité est restreinte aux pages d'un site et ainsi une analyse de complémentarité pourrait devenir envisageable dans le contexte du Web. Dans les cas, que nous croyons très rares, où un site posséderait un nombre anormalement élevé des pages, le système pourrait appliquer une analyse de complémentarité moins coûteuse, par exemple, on pourrait estimer la proximité structurelle entre deux pages à travers uniquement le calcul du facteur qui est fonction du chemin le plus court entre les deux pages (cf. terme S_{ij}^{spl} de l'équation 3.7) qui est associé à une complexité de $O(n^2)$, et donc ne pas utiliser les termes S_{ij}^{anc} et S_{ij}^{desc} qui amènent la compléxité à $O(n^4)$.

5.4 Perspectives

Nous avons évoqué dans le chapitre 4 que nous n'avions guère d'indications sur l'éventuelle représentativité des sites de la collection WT10g par rapport au Web. On peut donc imaginer que les résultats obtenus dans cette étude soient biaisés par rapport aux sites que nous avons testés — 641 sites parmi les 871 sites ayant au moins une page pertinente selon l'ensemble de jugements dont nous disposions. **Une première perspective** est de définir de critères pour caractériser les sites autres que le nombre de pages. Par exemple, de tels critères pourraient se baser sur la thématique du site et sur la structure du graphe sous-jacent au site (cf. degrés de stratification et compacité dans [Bot93]). Une fois les critères définis il faudrait sélectionner un sous-ensemble de sites hétérogènes par rapport aux critères, créer un ensemble de topiques et les jugements associés pour chaque site et finalement analyser le comportement de notre modèle sur ces sites. Cette étude permettrait de recueillir des indications sur l'influence des types de sites dans le modèle que nous proposons. Bien entendu les tâches de création des

topiques et des jugements qui leur sont associés peuvent s'avérer assez onéreuses. Alternativement, si nous identifions l'hétérogénéité de sites recherchée à l'intérieur d'une collection déjà existante, telle que WT10g, nous pouvons envisager de réutiliser cette collection. Pour ce qui concerne WT10g, d'après les observations que nous avons faites plus haut il faudrait (i) trouver plus de pages pertinentes par site aux topiques déjà existants (ii) créer d'autres topiques afin d'avoir un nombre raisonnable de topiques pour lesquels un site est pertinent. Il n'est cependant pas sûr que l'adaptation d'une collection déjà existante est moins coûteuse que la création d'une nouvelle collection.

Une deuxième perspective est la recherche d'autres facteurs structuraux qui servent d'indice de complémentarité entre deux pages d'un site. Comme nous avons vu plus haut, le calcul de la proximité structurelle constitue le goulot d'étranglement de notre algorithme, élevant sa complexité à $O(n^4)$ ou $O(n^4 \log n)$ selon que l'indexation utilise un seul niveau de contextes ou la hiérarchie de contextes. Plus précisément, les facteurs S_{ij}^{anc} et S_{ij}^{desc} sont les responsables de ces coûts si élevés. On pourrait donc envisager de rechercher d'autres facteurs qui donnent des indices sur la complémentarité de deux pages mais qui soient moins coûteux que ceux que nous venons d'indiquer. Une autre alternative à étudier consiste à continuer à utiliser les facteurs S_{ij}^{anc} et S_{ij}^{desc} mais faire de sorte que lorsque le site traité dépasse une certaine taille, le calcul des facteurs ne soit pas fait sur le graphe entier sous jacent au site mais sur un sous-graphe de taille réduite.

La troisième et dernière perspective est liée à l'utilisation de notre modèle dans le contexte d'une recherche dans le Web entier. Nous avons évoqué à la fin de la dernière section que l'application du modèle proposé dans cette étude au Web pourrait devenir envisageable si nous considérions qu'une unité logique d'information est composée uniquement d'information locale, c.-à-d. les pages complémentaires à une page donnée se trouvent dans le même site que cette page. Dans cette section, nous voulons évoquer une deuxième possibilité d'intégration de notre modèle de recherche locale à un modèle de recherche globale applicable au Web. Vu la croissance exponentielle du Web nous pensons qu'un modèle de recherche en deux étapes pourrait s'avérer plus performant que le modèle de recherche en une seule étape utilisé aujourd'hui. Les moteurs de recherche exhaustifs existant aujourd'hui (*Altavista*, *Google*, *eXcite*, etc.) ne sont plus capables de suivre la croissance du nombre de pages du Web. De plus nous ne voyons pas de raisons pour que cet état de fait change dans l'avenir proche. Nous pensons donc qu'une réduction de la dimensionnalité du Web pourrait être utile dans le processus de recherche d'information. Par exemple nous pourrions imaginer le Web comme étant un ensemble de sites plutôt que comme un ensemble de pages. C'est là où le modèle de recherche en deux étapes que nous avons évoqué plus haut prend sa place. Dans une première étape il s'agirait de localiser des sites (et non pas des pages) censés contenir de l'information liée à un certain thème. Une fois le(s) site(s) identifié(s), on passerait à une deuxième étape où la recherche s'effectuerait à l'intérieur de ce(s) site(s). C'est dans la deuxième étape qu'un modèle de moteur tel que celui que nous avons proposé dans cette étude pourrait être utilisé.

Plusieurs questions apparaissent lorsque l'on envisage un tel modèle de recherche : comment indexer les sites ? tous les sites devraient-ils posséder un moteur de recherche local ? les deux étapes du processus de recherche pourraient-elles être invisibles à l'utilisateur final ? Discutons brièvement ces questions. Pour ce qui est de l'indexation nous avons commencé

à examiner des approches d'indexation de sites basés sur les liens. Une première approche possible consiste à ne pas utiliser le contenu de toutes les pages d'un site pour indexer un site mais seulement le contenu de certaines pages que nous appelons *pages représentatives du site.* Afin de les identifier une possibilité serait d'exploiter la structure sous-jacente au graphe du site. Une autre approche possible consisterait à identifier les liens inter-sites qui pointent sur le site que nous voulons indexer et utiliser le texte de l'ancre de ces liens pour générer l'index du site. Cette approche peut ne pas se révéler très utile si les liens inter-site ont tendance à pointer sur des pages particulières d'un site, e.g. la page d'accueil, qui peuvent ne pas avoir un contenu représentatif des thèmes traités par le site.

Par rapport à la deuxième question, il est sûr que dans un modèle de recherche en deux étapes l'idéal serait que tous les sites possèdent un moteur de recherche local. Nous ne connaissons pas de statistiques qui estiment le pourcentage de sites dans le Web disposant d'un moteur de recherche local étant bien entendu qu'un nombre non-négligeable de sites ne possèdent pas de tels moteurs. Si les moteurs de recherches exhaustifs adoptent l'attitude de ne plus indexer les pages des sites mais d'indexer seulement les sites d'une manière globale, une certaine pression pourrait se placer du côté des concepteurs de site pour qu'ils installent des moteurs de recherche locaux. Il est vrai que l'utilisateur dispose toujours du mécanisme de navigation pour récupérer de l'information pertinente à l'intérieur d'un site. Cependant, vu le temps qu'il risque d'investir dans la navigation dans le site avant de trouver une information pertinente il préférera vraisemblablement utiliser un moteur de recherche indexant des pages qui lui fournira rapidement des réponses même si c'est au détriment de la qualité. Pour ce qui est de la troisième question, afin que la division du processus de recherche en deux étapes soit invisible à l'utilisateur, il faudrait évidemment (i) que les langages de requêtes des moteurs de recherche de sites et des moteurs locaux soient compatibles et (ii) qu'un protocole de communication existe entre les deux types de moteur de recherche. Nous pensons que cela ne sera possible que dans le cas où un organisme ayant une influence importante sur le développement des technologies Web tel que le consortium *w3c* deviendrait l'avocat de telles idées.

Annexe A

Exemple de document TREC

Une partie du document WTX001-B19-274 :

```
<DOC>
<DOCNO>WTX001-B19-274</DOCNO>
<DOCOLDNO>IA001-000003-B018-214</DOCOLDNO>
<DOCHDR>
http://search.nerdworld.com:80/nw147.html 199.94.217.62 19970101040619
text/html 78160
HTTP/1.0 200 OK
Server: Netscape-Communications/1.1
Date: Wednesday, 01-Jan-97 04:10:01 GMT
Last-modified: Tuesday, 24-Dec-96 17:33:06 GMT
Content-length: 77967
Content-type: text/html
</DOCHDR>
<HTML>
<HEAD>
<TITLE>Nerd World : REAL ESTATE</TITLE>
<! ***ADINFO***  CATEGORY=147;PARENT=17; !>
<META Name="description" Content="Large Index of REAL ESTATE related
internet resources created by Nerd World Media.
One of 0 categories within the Nerd World Internet Subject Index and Search
tool.  List your resource or Create your Own Index. All for Free!">
<META Name="keywords" Content="REAL ESTATE,CORPORATE REAL ESTATE,CUSTOM
HOMES,HOME INSPECTION,KNOWLEDGE,APPRAISAL,RENTAL,TIMESHARES,REAL ESTATE BY
REGION,REALTORS">
<BASE href="http://www.nerdworld.com/" TARGET="_parent">
</HEAD>
<FRAMESET COLS="30%,70%">
<NOFRAMES>
<BODY BGCOLOR=#FFFFFF>
<center><H1>REAL ESTATE<FONT SIZE=-1> BY NERD WORLD MEDIA<FONT
SIZE=-1>(TM)</FONT></FONT></H1>
<table border><tr>
<td><A HREF="about.html">What is Nerd World</A><br></td>
<td><A HREF="index.cgi">Nerd World Home Page</A><br></td>
<td><A HREF="nwadd.cgi">Add your Web Site</A><br></td>
<td><A HREF="nwerror.cgi">Report an Error</A><br></td>
</tr></table>
```

```
[...]


</BODY>
</NOFRAMES>
<FRAME SRC="cgi-bin/vcat.cgi?147" MARGINWIDTH=1 MARGINHEIGHT=1>
<FRAME SRC="cgi-bin/vdata.cgi?147" MARGINWIDTH=1 MARGINHEIGHT=1>
</FRAMESET>
</HTML>

</DOC>
```

Annexe B

Exemples de topiques TREC

Le topique numéro 458 :

```
<top>
<num> Number: 458
<title> fasting

<desc> Description:
Find documents that discuss fasting for religious reasons.

<narr> Narrative:
A relevant document discusses fasting as related to
periods of religious significance.  Relevant documents
should state the reason for fasting and the benefits to
be derived.

</top>
```

Le topique numéro 459 :

```
<top>

<num> Number: 459
<title> when can a lender foreclose on property

<desc> Description:
Identify documents which state the circumstances
under which a lender can legally foreclose on a
property.

<narr> Narrative:
A relevant document will include the statutes
under which a lender can foreclose on a property,
the appeal process available to the property
owner, the maximum amount the property can be sold
for and any mitigating or legal options available
```

```
to the property owner to stall the sale by the
lender.

</top>
```

Annexe C

La liste des mots vides

a	behind	every	has	let	number	puts	such	very
about	being	everybody	have	lets	numbers	q	sure	w
above	beings	everyone	having	like	o	quite	t	want
across	best	everything	he	likely	of	r	take	wanted
after	better	everywhere	her	long	off	rather	taken	wanting
again	between	f	here	longer	often	really	than	wants
against	big	face	herself	longest	old	right	that	was
all	both	faces	high	m	older	right	the	way
almost	but	fact	high	made	oldest	room	their	ways
alone	by	facts	high	make	on	rooms	them	we
along	c	far	higher	making	once	s	then	well
already	came	felt	highest	man	one	said	there	wells
also	can	few	him	many	only	same	therefore	went
although	cannot	find	himself	may	open	saw	these	were
always	case	finds	his	me	opened	say	they	what
among	cases	first	how	member	opening	says	thing	when
an	certain	for	however	members	opens	second	things	where
and	certainly	four	i	men	or	seconds	think	whether
another	clear	from	if	might	order	see	thinks	which
any	clearly	full	important	more	ordered	seem	this	while
anybody	come	fully	in	most	ordering	seemed	those	who
anyone	could	further	interest	mostly	orders	seeming	though	whole
anything	d	furthered	interested	mr	other	seems	thought	whose
anywhere	did	furthering	interesting	mrs	others	sees	thoughts	why
are	differ	furthers	interests	much	our	several	three	will
area	different	g	into	must	out	shall	through	with
areas	differently	gave	is	my	over	she	thus	within
around	do	general	it	myself	p	should	to	without
as	does	generally	its	n	part	show	today	work
ask	done	get	itself	necessary	parted	showed	together	worked
asked	down	gets	j	need	parting	showing	too	working
asking	down	give	just	needed	parts	shows	took	works
asks	downed	given	k	needing	per	side	toward	would
at	downing	gives	keep	needs	perhaps	sides	turn	x
away	downs	go	keeps	never	place	since	turned	y
b	during	going	kind	new	places	small	turning	year
back	e	good	knew	new	point	smaller	turns	years
backed	each	goods	know	newer	pointed	smallest	two	yet
backing	early	got	known	newest	pointing	so	u	you
backs	either	great	knows	next	points	some	under	young
be	end	greater	l	no	possible	somebody	until	younger
became	ended	greatest	large	nobody	present	someone	up	youngest
because	ending	group	largely	non	presented	something	upon	your
become	ends	grouped	last	noone	presenting	somewhere	us	yours
becomes	enough	grouping	later	not	presents	state	use	z
been	even	groups	latest	nothing	problem	states	used	
before	evenly	h	least	now	problems	still	uses	
began	ever	had	less	nowhere	put	still	v	

Annexe D

Exemple de jugements TREC

Une partie du fichier de jugements concernant les topiques 458 et 459 :

```
[...]
458 0 WTX103-B11-515 0
458 0 WTX103-B17-104 0
458 0 WTX103-B17-74 0
458 0 WTX103-B18-145 0
458 0 WTX103-B21-205 0
458 0 WTX103-B22-170 0
458 0 WTX103-B30-37 0
458 0 WTX103-B33-379 0
458 0 WTX103-B34-7 0
458 0 WTX103-B39-20 0
458 0 WTX103-B40-105 0
458 0 WTX103-B40-249 0
458 0 WTX103-B46-196 0
458 0 WTX103-B47-254 0
458 0 WTX103-B49-20 0
458 0 WTX103-B50-65 0
459 0 WTX001-B04-89 0
459 0 WTX001-B18-140 0
459 0 WTX001-B19-274 0
459 0 WTX001-B35-457 0
459 0 WTX001-B36-176 0
459 0 WTX001-B37-26 0
459 0 WTX001-B37-36 0
459 0 WTX001-B45-396 0
459 0 WTX002-B14-96 0
459 0 WTX002-B15-146 0
459 0 WTX002-B15-150 1
459 0 WTX002-B17-21 0
459 0 WTX002-B17-23 0
459 0 WTX002-B17-27 0
459 0 WTX002-B17-39 0
459 0 WTX002-B17-41 1
459 0 WTX002-B17-45 1
459 0 WTX002-B17-54 0
459 0 WTX002-B34-146 0
459 0 WTX002-B38-191 0
[...]
```

Annexe E

La liste des sites testés

site_id	machine_hôte:port	(topic_id/nombre de pages pertinentes dans le site), ...
S7	152.157.32.62:80	T452/2
S9	160.133.244.200:80	T490/1
S22	1upmc-isd-001.isdip.upmc.edu:80	T487/2 T492/1
S36	205.130.176.100:80	T452/2
S75	a.aok.com:80	T476/1
S102	abqcvb.org:80	T493/1
S137	acupuncture.com:80	T454/1
S191	alterworld.com:80	T494/48
S202	ammi.com:80	T496/3
S212	ananse.tor.hookup.net:80	T474/1
S223	antique-shop.com:80	T472/1
S231	aone.com:80	T452/1
S247	aptspotlight.com:80	T493/2
S256	archatl.com:80	T453/1
S265	arsdata.com:80	T462/3
S329	barnowl.lanl.gov:80	T500/2
S333	bbs.pconnect.com:80	T493/1
S469	ccme-mac4.bsd.uchicago.edu:80	T454/1
S471	cdnet.com:80	T474/1
S489	chem.iupui.edu:80	T500/1
S517	commlink.atlantic.county.lib.nj.us:80	T465/1
S531	condor.sccs.swarthmore.edu:80	T453/1
S558	cyberday.com:80	T474/1
S587	dem0nmac.mgh.harvard.edu:80	T454/13 T487/1
S605	diogenes.oplin.lib.oh.us:80	T468/1
S606	disaster.org:80	T453/1
S609	dli-www.army.mil:80	T490/1
S622	doyle.madison.k12.wi.us:80	T452/1
S623	dropd.com:80	T494/2 T495/2
S639	easydiet.com:80	T451/1
S640	ec3.earthchannel.com:80	T454/1 T490/1
S642	ecngate.ecn.co.uk:80	T484/2
S649	edison-ford-estate.com:80	T468/2
S657	emernet.emergency.com:80	T468/1 T480/1
S661	energy.aste.usu.edu:80	T452/2
S663	english-www.hss.cmu.edu:80	T494/4
S682	finearts.com.au:80	T475/1
S708	gearhead.com:80	T478/1 T489/1
S710	geffen.com:80	T494/4
S719	geo.arc.nasa.gov:80	T465/4
S726	gi2.mse.cwru.edu:80	T463/1
S729	gimli.whoi.edu:80	T494/1
S776	gw.dncc96.org:80	T478/2 T495/1
S788	helix.com:80	T454/1
S854	ibforecast.com:80	T451/1
S856	ibogaine.desk.nl:80	T454/1

S864	ilcommerce.com:80	T457/2
S878	indy.afroam.org:80	T455/2 T479/1 T495/1
S893	insiderreports.com:80	T492/2
S905	internet-mall.com:80	T451/1 T462/4 T467/1 T469/2 T472/4 T475/1 T496/1
S925	ite.io.com:80	T490/2
S927	itl1.itlnet.net:80	T463/1
S966	jumper.mcc.ac.uk:80	T460/1
S981	kiaskus.net.gull-lake.sk.ca:80	T452/1
S1018	lacebark.ntu.edu.au:80	T475/1
S1024	land.net:80	T462/1
S1025	landtrust.org:80	T452/3
S1072	ljswc.ucsd.edu:80	T488/1
S1082	longmont.com:80	T493/1
S1119	mail.leaderu.com:80	T460/1 T488/1
S1123	mailgate.partners-in-technology.co.uk:80	T463/1
S1150	market.boulder.co.us:80	T457/1
S1185	mdsg.umd.edu:80	T491/1
S1191	mediacastle.com:80	T483/1
S1206	merchantmall.com:80	T474/1
S1207	merck.utulsa.edu:80	T476/1
S1213	messiah.simplenet.com:80	T458/1
S1251	mmgco.com:80	T474/1
S1266	monomer.psrc.usm.edu:80	T475/1
S1288	multicheck.com:80	T474/1
S1322	networx.on.ca:80	T451/1
S1329	newton.math.grin.edu:80	T476/1
S1337	nightlight.com:80	T474/1
S1408	online96.com:80	T462/7
S1419	operon.com:80	T500/1
S1422	opus.co.tt:80	T492/3
S1503	prc-wwwserv.idap.indiana.edu:80	T490/1
S1522	ptr.ok-connect.com:80	T493/5
S1525	pubs.acs.org:80	T454/1
S1538	rainfrog.com:80	T458/1 T467/1
S1550	relaunch.com:80	T474/1
S1553	renne.lib.montana.edu:80	T452/1
S1575	rushdie.kcpl.lib.mo.us:80	T452/1 T495/1
S1600	scotland.bae.uga.edu:80	T490/1
S1601	scotland.rampant.com:80	T463/3
S1605	scratchy.hcrhs.hunterdon.k12.nj.us:80	T461/1
S1614	sd48.mountain-inter.net:80	T452/1
S1623	seagrant.orst.edu:80	T452/1
S1632	search.shareware.com:80	T499/1
S1636	seattlemusicweb.com:80	T494/1
S1655	server.uofdhigh.k12.mi.us:80	T452/1
S1686	shiva.uoregon.edu:80	T452/2
S1704	simon.com:80	T472/1
S1708	single.kaist.ac.kr:80	T475/1
S1718	siteit.com:80	T452/1
S1738	smartdev.com:80	T494/1
S1757	socorro.k12.tx.us:80	T468/1
S1806	starnews.wilmington.net:80	T457/5
S1833	sturgeon.irm1.r2.fws.gov:80	T470/1
S1858	sustainable.org:80	T452/1
S1901	telluridemm.com:80	T452/2
S1904	tenagra.com:80	T474/1
S1906	texasfishing.com:80	T488/2
S1952	toocool.com:80	T451/1
S1981	tuthmosis.adone.com:80	T476/1 T495/1
S1993	un.kiev.ua:80	T452/1
S1998	unixwannabe.lanminds.com:80	T459/1
S2034	vh0806.infi.net:80	T452/1
S2080	water.swc.state.nd.us:80	T452/1
S2107	wetland.usace.mil:80	T452/1
S2116	willow.ncfes.umn.edu:80	T452/1 T482/1
S2121	wiseacres.prodigy.com:80	T494/1
S2129	wolfden.swsc.k12.ar.us:80	T452/1
S2150	wtbradley.com:80	T465/1
S2168	wwitch.unl.edu:80	T452/1

S2197	www.1realestate.com:80	T462/1
S2241	www.aaacu.com:80	T492/1
S2255	www.aarp.org:80	T454/1 T492/2
S2267	www.abate.com:80	T490/2
S2297	www.abqcvb.org:80	T493/1
S2332	www.accessrex.com:80	T452/6 T457/2 T499/1
S2353	www.aclu.org:80	T455/3 T464/5 T478/2 T495/4
S2379	www.activemed.com:80	T454/3
S2422	www.adpak.com:80	T474/1
S2460	www.aenet.org:80	T491/1
S2500	www.aion.demon.co.uk:80	T463/2
S2519	www.alaskana.com:80	T452/1 T476/1
S2520	www.alasu.edu:80	T479/1
S2526	www.alexanderlaw.com:80	T457/1 T498/1
S2542	www.alliance-dc.org:80	T468/1
S2550	www.allny.com:80	T472/1
S2578	www.am-osteo-assn.org:80	T454/1
S2579	www.amagazine.com:80	T455/1 T476/1
S2592	www.americaiidirect.com:80	T474/1
S2609	www.amotion.com:80	T457/1
S2664	www.appli.com:80	T493/2
S2710	www.ariel.com.au:80	T463/1
S2753	www.ashland.or.us:80	T459/2
S2760	www.aspca.org:80	T465/1 T467/10
S2762	www.aspenserv.com:80	T452/2
S2769	www.assocbank.com:80	T493/1
S2835	www.autobest.com:80	T457/1
S2836	www.autoboard.com:80	T457/1
S2840	www.autoplaza.com:80	T457/1
S2939	www.black-collegian.com:80	T479/1
S2991	www.bootnet.com:80	T468/1 T476/1
S3017	www.bracewel.demon.co.uk:80	T463/1
S3092	www.buchalter.com:80	T459/1
S3094	www.buckeyehomes.com:80	T459/1
S3117	www.bumpstop.com:80	T457/1
S3123	www.burgarrowe.com:80	T459/1
S3130	www.burrelles.inter.net:80	T457/3 T484/1 T494/1
S3135	www.bus.usu.edu:80	T454/1
S3163	www.buysmart.com:80	T452/1
S3170	www.bwanazulia.com:80	T471/1
S3182	www.bytethis.net:80	T494/1
S3198	www.c4support.bss.org:80	T495/1 T500/1
S3256	www.calicotech.com:80	T474/1
S3259	www.california-fsbo.com:80	T459/1
S3271	www.cals.wisc.edu:80	T465/1
S3281	www.camprh.com:80	T453/1
S3292	www.ccplustours.com:80	T490/1
S3309	www.cd-plus.com:80	T494/3
S3363	www.cei.org:80	T452/2 T495/1
S3382	www.cellustar.co.uk:80	T484/1
S3393	www.charity.org:80	T453/12
S3408	www.checkers-llp.com:80	T492/1
S3409	www.checkfree.com:80	T474/1
S3413	www.checkthenet.com:80	T470/1
S3428	www.chem.umn.edu:80	T475/2
S3443	www.chevrolets4less.com:80	T457/1
S3559	www.city-online.com:80	T465/1
S3565	www.cityfarmer.org:80	T453/1
S3571	www.ckf.org:80	T452/1 T488/1
S3574	www.claremont.org:80	T458/1
S3637	www.cmsenergy.com:80	T452/1
S3646	www.co.blm.gov:80	T452/4
S3649	www.co.leon.fl.us:80	T469/1
S3654	www.co.washington.or.us:80	T452/1 T459/2
S3657	www.coastaltraveler.com:80	T472/1
S3728	www.come2az.com:80	T452/1
S3762	www.commpro.com:80	T488/4 T495/1
S3768	www.community-care.org.uk:80	T454/1

S3770	www.communitynetwork.com:80	T457/1
S3825	www.computerbits.com:80	T476/1 T481/1 T496/2
S3889	www.consciouschoice.com:80	T454/1
S3900	www.constellation.org:80	T478/1
S3907	www.consultco.com.au:80	T474/1
S3912	www.contact.org:80	T453/1
S3954	www.copywriter.com:80	T474/6
S3970	www.corningcu.org:80	T492/1
S3989	www.cosmeticmall.com:80	T498/1
S3997	www.costarica.org:80	T484/1
S4016	www.countynet.com:80	T476/1
S4034	www.cove.com:80	T452/1
S4049	www.cpco.com:80	T452/1 T482/1
S4081	www.crca.ca:80	T452/2
S4141	www.ctj.org:80	T492/2 T495/1
S4152	www.ctunitedway.org:80	T453/1
S4154	www.ctxmort.com:80	T459/1
S4169	www.cumberlink.com:80	T494/1
S4174	www.curbet.com:80	T457/5
S4200	www.cuttingedge.org:80	T454/1 T478/1 T495/3
S4245	www.cybercruises.com:80	T452/1 T477/1
S4309	www.cyberstrip-live.com:80	T479/1
S4334	www.cyberxpress.com:80	T451/1
S4362	www.cyquest.com:80	T467/4
S4395	www.dallasfed.org:80	T492/1
S4473	www.dbqinc.com:80	T493/2
S4484	www.dcfd.com:80	T480/1
S4487	www.dciexpo.com:80	T474/1
S4511	www.dealsonwheels.com:80	T457/3
S4521	www.debbie.com:80	T471/1
S4534	www.deerbusters.com:80	T465/1 T480/1
S4573	www.denveronline.com:80	T472/1
S4609	www.destinationmaine.com:80	T452/3
S4628	www.dgi.ca:80	T463/1
S4630	www.dgm-online.com:80	T463/1
S4631	www.dgs.state.md.us:80	T478/1 T492/1
S4643	www.diablopubs.com:80	T452/1 T472/1 T495/1
S4724	www.dircon.co.uk:80	T463/1
S4753	www.disu.ibm.com:80	T474/1
S4801	www.doe-hbcu-massie-chair.com:80	T478/1 T479/2
S4803	www.doesgodexist.org:80	T452/1 T495/1
S4806	www.dogzone.com:80	T467/7
S4807	www.doh.wa.gov:80	T465/1
S4840	www.dotmusic.co.uk:80	T494/1
S4864	www.draving.com:80	T462/1 T495/1
S4885	www.drgreene.com:80	T465/1
S4909	www.druginfonet.com:80	T454/5 T496/1
S4927	www.dsp.ee.ic.ac.uk:80	T490/1
S5020	www.eatright.org:80	T453/1 T489/5
S5081	www.edintattoo.co.uk:80	T463/1
S5138	www.einet.net:80	T485/1
S5139	www.einet.net:8000	T485/1
S5161	www.elkhound.com:80	T467/1
S5166	www.ellsworth.af.mil:80	T490/1
S5214	www.entermag.com:80	T476/1
S5234	www.episcopalnet.org:80	T458/2
S5250	www.erabuxton.com:80	T459/1
S5267	www.esd.ornl.gov:80	T452/3 T500/1
S5269	www.esf.edu:80	T452/2 T470/1
S5312	www.evaa.org:80	T457/1
S5323	www.evergreen.ca:80	T452/1 T482/1
S5327	www.everybodys.org:80	T476/2 T494/1
S5347	www.execlend.com:80	T459/1
S5348	www.execreport.com:80	T452/1 T474/1 T492/1 T493/1 T496/1
S5369	www.expressmedia.co.uk:80	T463/3
S5403	www.failte.com:80	T474/1
S5418	www.familyanimal.com:80	T467/2
S5488	www.fi.edu:80	T468/1

S5538	www.flick.com:80	T474/1
S5542	www.floridaforeclosures.com:80	T459/1
S5577	www.freeminds.org:80	T456/1 T460/1 T495/1
S5581	www.freeprofit.com:80	T474/3
S5592	www.ftech.com:80	T463/1
S5605	www.gael-net.co.uk:80	T463/2
S5632	www.geistware.com:80	T479/1
S5665	www.gii.org:80	T474/1
S5671	www.ginn.co.uk:80	T468/1
S5683	www.glasgow-fair.co.uk:80	T463/3
S5690	www.glinn.com:80	T464/1 T478/1
S5697	www.globalauto.com:80	T457/2
S5711	www.gloria-brame.com:80	T474/1
S5728	www.gogirlmag.com:80	T483/1 T489/1 T490/1
S5746	www.golferspassport.com:80	T452/1
S5759	www.goodsam.org:80	T454/1
S5793	www.granta.demon.co.uk:80	T463/1
S5818	www.greaterunion.com.au:80	T476/1
S5824	www.greeksource.com:80	T479/8
S5864	www.gt.com:80	T453/1
S5892	www.gunnison.com:80	T452/1
S5897	www.gv.ex.state.ut.us:80	T452/2
S5899	www.gvrd.bc.ca:80	T452/2
S5919	www.halcyonyarn.com:80	T463/2
S6039	www.immedia.com.au:80	T476/2 T494/6
S6063	www.infolink.morris.mn.us:80	T452/1
S6072	www.infotoday.com:80	T474/1
S6106	www.innerself.com:80	T452/1
S6176	www.internet-ad.com:80	T457/1
S6177	www.internet-mall.com:80	T451/1 T462/3 T467/1 T469/2 T472/1 T497/4
S6184	www.internex.net:80	T474/1
S6213	www.investchoice.com:80	T492/1
S6292	www.ist.net:80	T462/4
S6318	www.itva.org:80	T463/1
S6459	www.karenpryor.com:80	T467/2
S6464	www.karlnagel.com:80	T474/1
S6503	www.kccomputeruser.com:80	T474/1
S6520	www.kctv.com:80	T479/1
S6527	www.kdux.com:80	T494/7
S6558	www.keokee.com:80	T452/1
S6565	www.kevinmitnick.com:80	T475/1
S6583	www.kgtv.com:80	T454/2 T496/1
S6602	www.kidsource.com:80	T465/2 T489/1
S6631	www.kingsley.com:80	T454/3
S6651	www.kissasylum.com:80	T494/1
S6736	www.koreainterad.com:80	T462/1
S6740	www.korn.demon.co.uk:80	T463/1
S6742	www.kortright.org:80	T452/1
S6748	www.kpmg.co.uk:80	T474/1
S6788	www.ksps.org:80	T469/1
S6841	www.laig.com:80	T483/2
S6847	www.lakeunion.com:80	T459/1
S6866	www.lasalle.com:80	T463/1 T469/1
S6880	www.lavamind.com:80	T452/1 T467/3
S6892	www.law.umkc.edu:80	T479/1
S6901	www.lawlead.com:80	T459/4 T490/1
S6919	www.lazer103.com:80	T453/1
S6929	www.lcci.com:80	T493/1
S6975	www.leeislandcoast.com:80	T468/1
S6976	www.leesburgva.com:80	T452/1
S6992	www.leginfo.state.sc.us:80	T479/1
S7033	www.lff.org:80	T479/1
S7039	www.lhdigest.com:80	T459/1 T478/1
S7041	www.lhsusa.com:80	T470/1
S7055	www.lib.umb.edu:80	T492/1
S7102	www.lionsgate.com:80	T454/1
S7107	www.livingplanet.org:80	T452/1
S7130	www.loanpage.com:80	T459/1

S7147	www.loe.org:80	T452/1 T454/2 T470/1
S7181	www.long-dog.com:80	T467/1
S7233	www.lsuc.on.ca:80	T459/1
S7254	www.lumber.com:80	T482/1
S7278	www.lwc.edu:80	T479/1
S7312	www.macdude.org:80	T463/4
S7324	www.macmediaweb.com:80	T452/1
S7345	www.mad-cow.org:80	T454/3 T465/1
S7355	www.madison.com:80	T452/1
S7571	www.marlborough.com:80	T493/1
S7576	www.maroon.com:80	T467/1
S7601	www.marylandnet.com:80	T478/1
S7605	www.masdef.org:80	T458/1
S7622	www.massport.com:80	T469/1
S7690	www.mbemag.com:80	T478/1
S7753	www.mcwilliams.com:80	T490/4 T495/3
S7773	www.me3.org:80	T452/1
S7782	www.med.cornell.edu:80	T454/1 T500/1
S7809	www.mediashower.com:80	T452/1 T458/1 T494/1
S7810	www.mediasyn.com:80	T453/1
S7829	www.medtronic.com:80	T454/8 T487/1
S7853	www.melville.org:80	T463/1
S7864	www.mena-ark.com:80	T492/1
S7878	www.mercynet.org:80	T454/1 T487/1
S7889	www.merrimack.org:80	T452/1
S7933	www.mfhs.edu:80	T454/1
S7950	www.mhs.mendocino.k12.ca.us:80	T463/2
S8126	www.modernmedicine.com:80	T454/2 T487/1 T498/1
S8156	www.moore-information.com:80	T452/1
S8169	www.mpmlaw.com:80	T492/1
S8191	www.municipalworld.com:80	T490/1
S8194	www.music-stop.co.uk:80	T494/1
S8213	www.n-net.com:80	T452/1
S8222	www.nahat.net:80	T487/1
S8226	www.nami.org:80	T454/3
S8228	www.naples-online.com:80	T493/1
S8230	www.nashvilleparent.com:80	T467/1
S8234	www.naughty.com:80	T476/1
S8271	www.net-link.com:80	T476/1
S8293	www.netoasis.com:80	T452/2
S8300	www.netpubsintl.com:80	T474/1
S8315	www.netstrider.com:80	T452/1
S8357	www.next-wave.net:80	T472/1
S8374	www.nhptv.org:80	T452/1
S8376	www.ni-inc.com:80	T452/1
S8382	www.nicosoft.com:80	T452/1
S8389	www.niicorp.com:80	T474/1
S8392	www.nirs.org:80	T452/1
S8412	www.nolopress.com:80	T459/1
S8416	www.noonanrusso.com:80	T454/1
S8429	www.now.org:80	T453/1
S8470	www.nylaarp.com:80	T455/1 T481/1
S8483	www.obee.com:80	T492/1
S8491	www.octitle.com:80	T459/1
S8514	www.omnibus.com:80	T463/1
S8535	www.orangecoast.com:80	T488/7 T495/1 T498/1
S8537	www.orchids.org:80	T497/3
S8603	www.pcis.net:80	T460/1
S8610	www.peacenet.com:80	T494/1
S8624	www.petsitters.com:80	T467/1
S8639	www.pibooks.com:80	T494/1 T495/1
S8647	www.pinecreek.freenet.mb.ca:80	T472/1 T495/1
S8651	www.pinkysplace.com:80	T476/1
S8663	www.pitre.com:80	T452/1
S8664	www.pix.za:80	T476/2
S8669	www.pixelmotion.ns.ca:80	T463/1
S8673	www.pjrcpa.com:80	T492/1
S8747	www.prairiemtn.mb.ca:80	T452/1

S8748	www.pratt.lib.md.us:80	T478/1
S8836	www.psrc.usm.edu:80	T475/1
S8840	www.psychiatry.ubc.ca:80	T454/1
S8848	www.ptla.org:80	T459/1
S8857	www.publicpurpose.com:80	T480/2
S8866	www.pulitzer.com:80	T453/1
S8867	www.pulpless.com:80	T474/1
S8882	www.qbsaul.demon.co.uk:80	T463/1
S8940	www.r1.fws.gov:80	T452/1 T465/1 T470/2 T482/1
S8957	www.radcyberzine.com:80	T463/1 T494/3
S8992	www.rainbird.com:80	T483/4
S8999	www.raindrops.com:80	T457/1 T475/1
S9009	www.raleigh.acn.net:80	T493/1
S9049	www.redhillstudios.com:80	T468/1
S9079	www.rehabnet.com:80	T467/1
S9091	www.reliefnet.org:80	T453/1 T495/1
S9106	www.renature.com:80	T460/1 T495/1
S9112	www.renonv.com:80	T486/3
S9113	www.renotahoe.com:80	T486/1
S9139	www.rih.com:80	T488/2
S9165	www.rls.org:80	T454/1
S9208	www.rocketvideo.com:80	T476/1
S9213	www.rocknews.com:80	T494/13
S9217	www.rockville.inter.net:80	T480/1
S9251	www.rosevilleautomall.com:80	T457/4
S9252	www.rosicrucian.com:80	T460/4 T495/4
S9257	www.rotary.ihost.com:80	T463/2
S9272	www.rowling.com:80	T492/1
S9289	www.rpinet.com:80	T453/1
S9320	www.rtci.com:80	T474/1
S9355	www.ruston.com:80	T493/1
S9366	www.rwjf.org:80	T487/1
S9403	www.safaar.com:80	T458/7
S9408	www.safe-t-child.com:80	T500/1
S9435	www.sailingbreezes.com:80	T452/1
S9494	www.sando.com:80	T459/3 T478/7
S9523	www.saranac.com:80	T452/1
S9545	www.saudhouse.com:80	T458/1
S9557	www.savingsbonds.com:80	T492/8
S9590	www.sbu.se:80	T487/1
S9633	www.schlager.com:80	T494/1
S9636	www.schnabel.com:80	T480/1
S9641	www.school.discovery.com:80	T468/1
S9659	www.sciam.com:80	T454/2 T468/1 T475/1 T495/1
S9660	www.scicomm.org.uk:80	T500/1
S9682	www.scoi.com:80	T475/1
S9687	www.scot.demon.co.uk:80	T463/1
S9688	www.scotch.com:80	T463/1
S9693	www.scotsgay.co.uk:80	T463/1
S9760	www.seconn.com:80	T469/1
S9772	www.seedman.com:80	T454/1 T470/1
S9775	www.seemall.com:80	T474/2
S9803	www.sequana.com:80	T500/2
S9811	www.sergeants.com:80	T465/2 T467/2
S9866	www.sfchamber.com:80	T493/1
S9868	www.sfcu.org:80	T459/1
S9902	www.shareholder.com:80	T453/1
S9912	www.shca.org:80	T467/1
S9925	www.sherryart.com:80	T452/1
S9928	www.shestheone.com:80	T476/1
S9929	www.shetland-news.co.uk:80	T463/1
S9942	www.shootingsports.com:80	T452/4
S9961	www.simplenet.com:80	T476/1
S10021	www.skicentral.com:80	T452/4
S10023	www.skifun.com:80	T452/1
S10034	www.skinews.com:80	T452/3
S10080	www.smartravel.com:80	T488/2
S10112	www.smpcollege.com:80	T453/1

S10137	www.snl.co.uk:80	T463/1
S10146	www.snp.org.uk:80	T463/1
S10164	www.socalvillage.com:80	T488/1
S10242	www.solutions.net:80	T452/1 T463/1 T490/1 T491/2 T495/1
S10242	www.solutions.net:80	T452/1 T490/1 T491/2 T495/1
S10245	www.somatics.com:80	T496/1
S10261	www.sonomazone.com:80	T497/1
S10308	www.southerncalifornia.com:80	T488/1
S10356	www.spcc.com:80	T452/1 T495/5
S10359	www.speaking.com:80	T468/1 T474/1 T495/2
S10397	www.sph.emory.edu:80	T454/1
S10402	www.spidacom.co.uk:80	T463/1
S10423	www.splitfire.com:80	T457/2
S10440	www.sports-medicine.com:80	T454/1
S10526	www.starcreations.com:80	T476/1
S10557	www.steamboatweb.com:80	T452/1
S10572	www.stercomm.com:80	T474/1
S10577	www.steve-hatfield.com:80	T459/1
S10595	www.stmichael.org:80	T458/3 T460/1
S10631	www.str.org:80	T490/1
S10658	www.stretcher.com:80	T467/2
S10693	www.styo.com:80	T459/1
S10702	www.succeed.com:80	T467/1
S10722	www.summitmed.com:80	T454/1
S10774	www.surfpac.navy.mil:80	T491/1 T495/1
S10795	www.suzton.com:80	T451/1
S10822	www.swms.com:80	T494/1
S10854	www.tactica.com:80	T452/1
S10860	www.tahoecountry.com:80	T452/1 T472/1
S10870	www.tamebeast.com:80	T467/2
S10882	www.tasc.ac.uk:80	T458/2 T484/1
S10892	www.taxlogic.com:80	T492/1
S10917	www.teelfamily.com:80	T452/1
S10930	www.telemedical.com:80	T490/2
S10959	www.theatlantic.com:80	T452/1 T456/1 T459/1 T465/1 T468/2 T478/1 T480/1 T495/25
S10974	www.thoracic.org:80	T454/1
S11014	www.tnef.com:80	T476/1
S11015	www.tnews.com:80	T452/1 T495/1
S11032	www.tomgill.com:80	T457/1
S11049	www.totalinc.com:80	T459/1
S11059	www.tourism.wa.gov:80	T452/1
S11069	www.tpl.lib.wa.us:80	T452/1
S11091	www.transcore.com:80	T498/1
S11092	www.transdat.com.au:80	T457/1
S11106	www.travel-michigan.state.mi.us:80	T452/1
S11123	www.travelspots.com:80	T463/1
S11179	www.tsi.it:80	T456/1 T462/1 T492/1
S11233	www.victoria.net:80	T452/2
S11234	www.victoriana.com:80	T463/1 T495/1
S11272	www.visitmaine.com:80	T452/1
S11278	www.vitapro.com:80	T453/5 T454/1
S11294	www.vonline.com:80	T452/2 T490/1
S11309	www.vwgroovin.com:80	T452/1
S11334	www.washingtoncitypaper.com:80	T476/1
S11335	www.washingtonforests.com:80	T452/1
S11347	www.wavcentral.com:80	T476/1
S11366	www.webmaker.mci.com:80	T474/1
S11369	www.webpointers.com:80	T492/1
S11421	www.wilmington.org:80	T467/1
S11470	www.woodtv.com:80	T476/1
S11481	www.worldparts.com:80	T457/1
S11494	www.wpvitv.com:80	T479/1
S11496	www.writepage.com:80	T463/1 T495/1
S11541	www.xworld.com:80	T494/2
S11551	www.ydr.com:80	T494/2
S11579	www.yoyo.com:80	T476/2
S11622	www1.dogtech.com:80	T474/1
S11648	www88.metronet.com:80	T500/1

Bibliographie

[ACG91] M. Agosti, R. Colotti, and G. Gradenigo. A two-level Hypertext Retrieval Model for Legal Data. In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 316–325, 1991.

[ACM87] ACM. *Proceedings of the First ACM Conference on Hypertext*, 1987.

[AH00] L.A. Adamic and B.A. Huberman. Power-law distribution of the world wide web. *Science*, 24(287), 2000.

[Ami97] E. Amitay. *Hypertext: the importance of being different.* MSc Dissertation. Centre for Cognitive Science. The University of Edinburgh, 1997.

[Ami00] E. Amitay. Trends, fashions, patterns, norms, conventions... and hypertext too, 2000.

[And] Olivier Andrieu. Site abondance - comparatif de fonctionnalités des moteurs de recherche. Disponible sur http://www.yahoo.com/outils/comparatif.html (dernier accès 01/07/2001).

[And73] M. R. Anderberg. *Cluster Analysis for Applications.* Academic Press, 1973.

[ATH00] Brian Amento, Loren Terveen, and Will Hill. Does "Authority" Mean Quality? Predicting Expert Quality Ratings of Web Documents. In *Proceedings of the ACM SIGIR conference*, 2000.

[BB99] K. Bharat and A. Broder. Mirror, Mirror on the Web: A Study of Host Pairs with Replicated Content. In *Proceedings of the 8th International WWW Conference*, 1999.

[BBDH00] Krishna Bharat, Andrei Z. Broder, Jeffrey Dean, and Monika Rauch Henzinger. A Comparison of Techniques to Find Mirrored Hosts on the WWW. *IEEE Data Engineering Bulletin*, 23(4):21–26, 2000.

[BCGM+99] Orkut Buyukkokten, Junghoo Cho, Hector Garcia-Molina, Luis Gravano, and Narayanan Shivakumar. Exploiting Geographical Location Information of Web Pages. In *Proc. of the ACM SIGMOD Workshop on the Web and Databases (WebDB'99)*, pages 91–96, 1999.

[BCH01] Peter Bailey, Nick Craswell, and David Hawking. Engineering a multi-purpose test collection for web retrieval experiments - draft* accepted. *Information Processing and Management*, 2001. (to appear).

[Bel00] R. K. Belew. *Finding Out About.* Cambridge Univ. Press, 2000.

[BKM+00] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph Structure in the Web. In *Proceedings of the Ninth International WWW Conference*. IW3C2, 2000.

[BL99] J. Borges and M. Levene. Data mining of user navigation patterns. In *Proceedings of the WEBKDD'99 workshop on Web Usage Analysis and User Profiling*, pages 31–36, San Diego, CA, USA, 1999.

[Bot93] R. A. Botafogo. Cluster analysis for hypertext systems. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 116–125. ACM, 1993.

[BP98] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International WWW Conference*. IW3C2, 1998.

[Bra96] T. Bray. Measuring the Web. In *Proceedings of the Fifth International WWW Conference*. IW3C2, 1996.

[Bus45] Vannevar Bush. As We May Think. *The Atlantic Monthly*, 176(1):101–108, July 1945.

[BV92] P. D. Bruza and T. P. Van Der Weide. Stratified Hypermedia Structures for Information Disclosure. *The Computer Journal*, 35(3):208–220, 1992.

[BYRN99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, 1999.

[CDI98] Soumen Chakrabarti, Byron E. Dom, and Piotr Indyk. Enhanced Hypertext Categorization Using Hyperlinks. In Laura M. Haas and Ashutosh Tiwary, editors, *Proceedings of SIGMOD-98, ACM International Conference on Management of Data*, pages 307–318, Seattle, US, 1998. ACM Press, New York, US.

[CGMP98] Junghoo Cho, Hector García-Molina, and Lawrence Page. Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1–7):161–172, 1998.

[Cha97] Chandra Chekuri and Prabhakar Ragavan. Web Search Using Automatic Classification. In *Proceedings of the 6th International WWW Conference*, Santa Clara, USA, 1997.

[CHR01] Nick Craswell, David Hawking, and Stephen Robertson. Effective Site Finding Using Link Anchor Information. In *Proceedings of the ACM SIGIR conference*, 2001.

[CK97] Y. Chiaramella and A. Kheirbek. An Integrated Model for Hypermedia and Information Retrieval. In M. Agosti and A. Smeaton, editors, *Information Retrieval and Hypertext*, pages 139–178. Kluwer Academic Publishers, 1997.

[Cro82] David H. Crocker. RFC 822 - Standard for the Format of Arpa Internet Text Messages, 1982. Dept. of Electrical Engineering, University of Delaware, Newark.

[CvdBD99] Soumen Chakrabarti, Martin van den Berg, and Byron E. Dom. Focused Crawling: A New Approach for Topic-Specific Resource Discovery. In *Proceedings of the 8th International WWW Conference*, Toronto, Canada, 1999.

[Def77] D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977.

[DGS00] Junyan Ding, Luis Gravano, and Narayanan Shivakumar. Computing geographical scopes of web resources. In *26th International Conference on Very Large Databases, VLDB 2000*, Cairo, Egypt, 2000.

[DH99] Jeffrey Dean and Monika Henzinger. Finding Related Pages in the World Wide Web. In *Proceedings of the 8th International WWW Conference*, Toronto, Canada, 1999.

[Dyr97] Curtis E. Dyreson. A Jumping Spider to Index Concepts that Span Pages. Technical Report TR 97/05, Department of Computer Science, James Cook University, Townsville, Australia, 1997.

[Dyr98] C. E. Dyreson. A Jumping Spider: Restructuring the WWW Graph to Index Concepts that Span Pages. In *Proceedings of the Seventh International WWW Conference (Workshop on Reuse of Web Information)*, Melbourne, Australia, 1998. IW3C2.

[FB96] N. Freed and N. Borenstein. RFC 2045 - Multipurpose Internet Mail Extensions (MIME) Part One : Format of Internet Message Bodies, 1996.

[FBY92] W. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms.* Prentice-Hall, 1992.

[FIG+97] R. Fielding, U.C. Irvina, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. RFC 2068 - Hypertext Transfer Protocol – HTTP/1.1, 1997.

[GKR98] David Gibson, Jon M. Kleinberg, and Prabhakar Raghavan. Inferring Web Communities from Link Topology. In *UK Conference on Hypertext*, pages 225–234, 1998.

[Hag97] Eric Hagen. An Information Retrieval System for Performing Hierarchical Document Clustering . Technical Report PCS-TR97-318, Dartmouth College, Computer Science, Hanover, NH, May 1997.

[HFBYL92] D. Harman, E. Fox, R. Baeza-Yates, and W. Lee. Inverted Files. In W.B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms*, pages 241–263. Prentice Hall, 1992.

[HP93] M. Hearst and C. Plaunt. Subtopic structuring for full-length document access. In *Proceedings of the 21st Annual International ACM/SIGIR Conference*, pages 59–68, Pittsburgh, PA, US, 1993.

[HSDT99] Kenji Hatano, Ryouichi Sano, Yiwei Duan, and Katsumi Tanaka. An interactive classification of web documents by self-organizing maps and search engines. In *Database Systems for Advanced Applications, DASFAA*, pages 35–42, 1999.

[HVCB00] D. Hawking, E. Voorhees, N. Craswell, and P. Bailey. Overview of the Ninth Text REtrieval Conference (TREC-9), 2000.

[JW97] K. Sparck Jones and P. Willet. *Readings in Information Retrieval.* Morgan Kaufmann Publishers, Inc., 1997.

[Kes63] M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14:10–25, 1963.

[KF92] Susan K. Kinnell and Carl Franklin. Hypercard and hypertext : A new technology. In Allen Kent, editor, *Encyclopedia of Library and Information Science*, volume 49, pages 278–295. 1992. New York.

[KK99] N. Kobayashi and F. Kitagawa. Finding a page-set in the WWW document and its application to search engines, 1999.

[Kle98] Jon Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[Kow97] G. Kowalsky. *Information Retrieval Systems: Theory and Implementation.* Kluwer Academic Publishers, 1997.

[KRRT99] S. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the web for emerging cyber-communities. In *Proceedings of the 8th International WWW Conference*, 1999.

[KSN97] S. Kurohashi, N. Shiraki, and M. Nagao. A method for detecting important descriptions of a word based on its density distribution in text. *Transactions of Information Processing Society of Japan*, 38(4), 1997.

[KT00] Mei Kobayashi and Koichi Takeda. Information Retrieval on the Web. *ACM Computing Surveys*, 32(2):144–173, 2000.

[LC95] H. Le-Crosnier. L'hypertexte en réseau : repenser la bibliothèque. *Bulletin des bibliothèques de France*, 40(2):23–31, 1995.

[LCVA01] W.S. Li, K. Candan, Q. Vu, and D. Agrawal. Retrieving and organizing web pages by "information unit". In *Proceedings of the Tenth International WWW Conference*, Hong Kong, China, 2001.

[LG99a] S. Lawrence and C. L. Giles. Searching the Web: General and Scientific Information Access. *IEEE Communications*, 37(1), 1999.

[LG99b] Steve Lawrence and C. Lee Giles. Accessibility of Information on the Web. *Nature*, 400:107–109, 1999.

[LKV00] Wen-Syan Li, Okan Kolak, and Quoc Vu. Defining logical domains in a web site. In *Proceedings of the ACM Hypertext Conference*, 2000.

[Luh57] H.P. Luhn. A statistical approach to mechanised encoding and searching of library information. *IBM Journal of Research and Development*, 1:309–317, 1957.

[Mar73] I. V. Marshakova. Document coupling system based on references taken from science citation index. *Teknicheskaya Informatsiya*, 3(6), 1973.

[MBNL99] Sanjay Kumar Madria, Sourav S. Bhowmick, Wee Keong Ng, and Ee-Peng Lim. Research issues in web data mining. In *Proceedings of the first International Conference on Data Warehousing and Knowledge Discovery*, pages 303–312, 1999.

[MM97] A. Mendelzon and T. Milo. Formal models of web queries. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 134–143, 1997.

[MT99] Yoshiaki Mizuuchi and Keishi Tajima. Finding context paths for web pages. In *Proceedings of the ACM Hypertext Conference*, 1999.

[OML00] Hyo-Jung Oh, Sung Hyon Myaeng, and Mann-Ho Lee. A Practical Hypertext Categorization Method Using Links and Incrementally Available Class Information. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

[PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Libraries Working Paper, 1998. Disponible sur: `http://citeseer.nj.nec.com/page98pagerank.html` (dernier accès 30/08/2001).

[PH97] J. Palme and A. Hopmann. RFC 2068 - MIME E-mail Encapsulation of Aggregate Documents, such as HTML, 1997.

[PK98] John Punin and Mukkai S. Krishnamoorthy. WWWPal System - A System for Analysis and Synthesis of Web Pages. In *Proceedings of the WebNet 98 Conference*, Orlando, USA, November 1998.

[Por97] M.F. Porter. An algorithm for suffix striping. In K. Sparck Jones and P. Willet, editors, *Readings in Information Retrieval*, pages 313–316. Morgan Kaufmann Publishers, Inc., 1997.

[PPR96] Peter Pirolli, James Pitkow, and Ramana Rao. Silk from a Sow's Ear: Extracting Usable Structures from the Web. In *Proceedings of the International Conference on Human Factors and Computing Systems*, pages 118 – 125. ACM, 1996.

[RM00] Davood Rafiei and Alberto Mendelzon. What is this page known for? computing web page reputations. In *Proceedings of the 9th International WWW Conference*, Amsterdam, 2000.

[Rob77] S. E. Robertson. The Probability Ranking Principle in IR. *Journal of Documentation*, 33(4), 1977.

[Ron96] Ron Weiss and Bienvenido Velez and Mark A. Sheldon and Chanathip Namprempre and Peter Szilagyi and Andrzej Duda and David K. Gifford. Hypursuit: A hierarchical network search engine that exploits content-link hypertext clustering. In *Proceedings of the Seventh ACM Conference on Hypertext*, pages 180–193, Wahington, USA, 1996.

[RWB00] S. E. Robertson, S. Walker, and M. Beaulieu. Experimentation as a Way of Life: Okapi at TREC. *Information Processing & Management*, 1(36):95–108, 2000.

[RWHB+94] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at TREC-3. In *Proceedings of TREC-3*, 1994.

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

[Sav97] J. Savoy. *Information Retrieval and Hypertext (M. Agosti and A. Smeaton)*, chapter Citation Schemes in Hypertext Information Retrieval. Kluwer Academic Publishers, 1997.

[SM83] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill Book Co., New York, 1983.

[Sma73] H. G. Small. Co-citation in the scientific literature. *Journal of the American Society for Information Science*, (24):265–269, 1973.

[Sou98] Soumen Chakrabarti and Byron E. Dom and Prabhakar Ragavan and Srifhar Rajagopalan and David Gibson and Jon Kleinberg. Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text. In *Proceedings of the 7th International WWW Conference*, Brisbane, Australia, 1998.

[Tea95] Guy Teasdale. L'hypertexte : historique et applications en bibliothéconomie, November 1995. Disponible sur `http://www.fas.umontreal.ca/EBSI/cursus/vol1no1/teasdale.html` (dernier accès 15/10/2001).

[THA99] L. Teerven, W. Hill, and B. Amento. Constructing, Organizing, and Visualizing Collections of Topically Related Web Resources. *ACM Transactions on Computer-Human Interaction*, 6(1):67–94, 1999.

[THM+99] Keishi Tajima, Kenji Hatano, Takeshi Matsukura, Ryouichi Sano, and Katsumi Tanaka. Discovery and Retrieval of Logical Information Units in Web. In *Proceedings of the ACM Digital Library Conference*, 1999.

[TMKT98] Keishi Tajima, Yoshiaki Mizuuchi, Masatsugu Kitagawa, and Katsumi Tanaka. Cut as a Querying Unit for WWW, Netnews, and E-mail. In *Proceedings of the ACM Hypertext Conference*, 1998.

[VH00] E. Voorhees and D. Harman. Overview of the Ninth Text REtrieval Conference (TREC-9), 2000.

[Voo86] Ellen Marie Voorhees. *The Effectiveness and Efficiency of Agglomerative Hierarchic Clustering in Document Retrieval.* PhD thesis, Cornell University, 1986.

[vR79] C.J. van Rijsbergen. *Information Retrieval.* Butterworths, 2nd edition, 1979.

[Wei74] B. H. Weinberg. Bibliographic coupling : A review. *Information Storage and Retrieval*, 10:189–196, 1974.

Bibliographie personnelle

- **Chapitre dans un livre**
  - **F. Aguiar.** *Improvement of Web Search by the Identification of Contextual Information.* In *P.S.Szczepaniak, J. Segovia, J. Kacprzyk, L.A. Zadeh, editors, "Intelligent Exploration of the Web"*, Springer-Verlag, Heildberg, GERMANY, 2002.
- **Publications dans des revues avec comité de séléction**
  - **F. Aguiar, M. Beigbeder.** *Discovering the Context of WWW Pages to Improve the Effectiveness of Local Search Engines.* In *The 4th International Conference on Flexible Querying Answering Systems*, Warsaw, POLAND, October 2000. Paru dans *Flexible Query Answering Systems - Recent Advances, Physica-Verlag*, 2000.
- **Conférences internationales avec comité de lecture**
  - **F. Aguiar, M. Beigbeder.** *Improvement of Web Retrieval by the Use of Contextual Information of Pages.* In *Proceedings of 12th ACM International Conference on Hypertext and Hypermedia*, Århus, DENMARK, August 2001.
  - **F. Aguiar, M. Beigbeder.** *Des moteurs de recherche efficaces pour des systèmes hypertextes grâce aux contextes des noeuds.* In *TICE2000*, Troyes, FRANCE, October 2000.
  - **F. Aguiar, B. L. Doan, M. Beigbeder.** *Deducing the Context Hierarchy of Web Sites for Information Retrieval Systems.* In WebNet99, Honolulu, USA, October 1999.
  - **F. Aguiar, B. L. Doan, M. Beigbeder.** *Dealing with Structured Documents in Information Retrieval Systems.* In WebNet99, Honolulu, USA, October 1999.
- **Rapport de recherche**
  - **F. Aguiar, S.H. He.** *Adapting Hierarchical Clustering Methods to Web Site Clustering.* Rapport de recherche, SIMMO, École Nationale Supérieure des Mines de St. Etienne, FRANCE, May 2002.

ÉCOLE NATIONALE SUPÉRIEURE
DES MINES DE SAINT-ETIENNE

UNIVERSITÉ JEAN MONNET
DE SAINT-ETIENNE

N° d'ordre : 272 ID

Nom : CARVALHO DE AGUIAR Fernando Jorge
Spécialité : Informatique
Titre : Modélisation d'un système de recherche d'information pour les systèmes hypertextes. Application à la recherche d'information sur le World Wide Web.

Résumé :
Dans un hypertexte, un document est souvent composé de plusieurs nœuds et non pas d'un seul. L'information véhiculée par un nœud donné peut difficilement être appréhendée à travers la lecture du seul contenu de ce nœud, le contenu des autres nœuds qui composent un document avec le premier nœud lui apportent un contexte. La connaissance de ce contexte est fondamentale dans la compréhension de l'information véhiculée par le premier nœud.

Un système de recherche d'information, ou plus couramment un moteur de recherche, appliqué au système hypertexte que constitue le Web devrait considérer dans son fonctionnement la fragmentation des documents hypertextuels en plusieurs pages : une page ne constitue pas un document à part entière, elle n'en est qu'une partie. Ainsi, pour bien indexer une page le contexte de l'information qu'elle véhicule doit être considéré. Les moteurs de recherche considérent souvent une page comme un document et l'indexent en analysant uniquement son contenu. Le contexte des pages est ignoré.

Dans ce travail nous proposons un modèle de recherche d'information pour un moteur de recherche appliqué à un système hypertexte constitué par un site Web. Ce modèle repose sur la construction d'un index à deux niveaux pour chacune des pages du site : un premier niveau, niveau inférieur, construit à partir du seul contenu de la page, et un deuxième niveau, niveau supérieur, construit à partir du contenu des pages qui apportent un contexte au contenu de la page en train d'être indexée. En améliorant la qualité des index des pages on cherche à améliorer l'efficacité du moteur de recherche.

Grâce à l'implémentation d'un prototype de moteur de recherche intégrant le modèle proposé ainsi que l'utilisation de la collection de tests WT10g issue des conférences TREC et adaptée à nos besoins, nous avons pu mener des expérimentations. Les résultats de ces dernières, une amélioration dans la qualité des réponses retournées par le moteur prototype, sont des indicateurs favorables de l'utilité de l'information contextuelle des pages. L'efficacité du moteur prototype a été comparée avec celle d'un moteur de recherche adoptant un modèle traditionnel où un seul niveau d'index, celui issu du seul contenu des pages, est utilisé.

Mots clés :
Hypertexte, recherche d'information sur le Web, moteur de recherche, graphe du Web, analyse de liens, bibliométrie, classification automatique