Recherche ciblée de documents sur le web

Amar-Djalil MEZAOUR

LRI, Université Paris Sud, 91405 Orsay Cedex
mezaour@lri.fr, http://www.lri.fr/∼mezaour

Résumé. Les langages de requêtes mots-clés pour le web manquent souvent de précision lorsqu'il s'agit de rechercher des documents particuliers difficilement caractérisables par de simples mots-clés (exemple : des cours java ou des photos de formule 1). Nous proposons un langage multi-critères de type attribut-valeur pour augmenter la précision de la recherche de documents sur le web. Nous avons expérimentalement montré le gain de précision de la recherche de documents basé sur ce langage.

1 Introduction

De par sa croissance et son développement, le web représente aujourd'hui une source importante de données hétérogènes (news, articles, photos, vidéos...). Les informations y sont stockées sous forme de documents identifiés d'une manière unique par urls et reliés entre eux par des liens hypertextes. Rechercher ou consulter une information particulière consiste à retrouver les urls des documents susceptibles de la contenir. Les moteurs de recherche ont été développés pour offrir aux utilisateurs des outils simples, mais néanmoins puissants, pour rechercher des documents sur le web. Un moteur de recherche (ex Google [Google, 2003]. . .) se décompose grossièrement en deux parties : un index web et un langage de requêtes utilisateur. L'index peut être vu comme un immense entrepôt de données où les documents webs sont stockés et indexés par mots clés après avoir été rapatriés par un robot explorateur. Un langage de requêtes mots-clés est proposé aux utilisateurs pour interroger l'index et accéder aux documents web qu'il contient. Pour cela, l'utilisateur spécifie une requête dans laquelle il précise l'ensemble des mots-clés caractérisant, selon lui, le ou les documents à rechercher. Cet ensemble de mots clés est soumis à l'index afin de retrouver les urls de documents contenant le plus d'occurences de ces mots. Les réponses renvoyées sont généralement très nombreuses, peu précises et ne correspondent pas nécessairement aux pages souhaitées par l'utilisateur. Il y a à cela deux raisons majeures. D'une part, le pouvoir expressif des requêtes d'un langage mots-clés ne permet pas de cerner avec exactitude les pages souhaitées. En effet, une requête mots-clés est limitée à la spécification des mots pertinents que doivent contenir les pages pour être considérées comme réponses, sans autre possibilité de décrire d'autres caractéristiques d'une page. Ainsi, pour la recherche de documents à faible contenu textuel (exemple : images, pdf. . .) ou pour la recherche de documents caractérisables autrement que par des mots clés (exemple : cours java ou c++), les requêtes mots-clés se montrent inappropriées. D'autre part, l'approche même qui considère toute page contenant les mots clés fournis dans une requête comme pertinente, sans tenir compte de la localisation de leur présence ni de la structure du document ni de son contexte accentue d'avantage l'imprécision des réponses. Par exemple, pour une recherche de documents de cours en c++, l'utilisateur soumet naïvement la requête « cours c++ » à un moteur de recherche sans autres alternatives pour décrire des cours c++. Le moteur de recherche renvoie en réponses quelques cours c++ mais aussi des documents décrivant l'emploi du temps des cours c++ dispensés dans une université ou les urls des forums traitant de c++. Dans cet article, nous montrons qu'une façon d'augmenter la précision de la recherche sur le web consiste à cibler la recherche de mots-clés en tenant compte de la structure de la page (ex son titre) ainsi que de son contexte (les liens entrant et sortant des pages voisines dans le graphe du web). Cette recherche mots-clés ciblée est combinée, à l'aide d'opérateurs logiques, avec d'autres critères concernant le type du document, son url . . .. Nous avons pour cela défini et implanté un langage de requêtes multi-critères dont nous avons évalué expérimentalement la précision.

Dans la section suivante, nous présentons un état de l'art des travaux existants liés à la recherche de documents sur le web. La section 3 est consacrée à la définition de la syntaxe et la sémantique des requêtes du langage que nous proposons. Nos expérimentations (protocole expérimental et évaluations) sont détaillées dans la section 4 de l'article. Nous terminons par quelques perspectives d'utilisation de notre langage de requêtes.

2 État de l'art

Les moteurs de recherche reposent dans leur majorité sur des langages mots-clés exécutables sur de gros index de pages web. Pour répondre à une requête mots-clés q, un moteur de recherche regroupe dans ses réponses les pages possédant le plus d'occurences des mots de q dans leur contenu (indifféremment de la localisation de ces mots). Un affinement supplémentaire est appliqué à l'aide de systèmes de classement (*PageRanking* [Brin et Page, 1998], source & authority [Kleinberg, 1999]) pour présenter en priorité les réponses jugées les plus intéressantes. Les pages pertinentes ne figurent pas nécessairement en grand nombre parmi ces réponses. Ceci s'explique par une caractérisation insuffisante des pages souhaitées par simple mots-clés sans tenir compte du contexte, de l'environnement, du type et de la structure d'un document. Google propose, en plus de sa recherche mots-clés classique, une option de recherche avancée qui permet à un utilisateur de cibler sa requête sur des parties bien précises d'une page. Avec cette option, le pouvoir expressif d'une requêtes est accru en permettant de cibler différentes parties d'une page (titre, mots de l'url. . .). L'utilisateur ne peut cependant pas combiner plusieurs requêtes sur différentes parties d'une page (cibler le titre et l'url en même temps par exemple).

D'autres approches proposent d'accroître la précision des langages requêtes mots-clés en enrichissant leurs requêtes par de nouveaux mots-clés inferés d'ontologies (spécifiques à un domaine ou générales de type WordNet [Felbaum, 1998, Voorhees, 1998]). Les nouveaux mots-clés ajoutés représentent souvent des synonymes ou des généralisations (en terme de concepts) possibles des mots-clés de la requête initiale conformément aux ontologies utilisées. *THESUs* [Halkidi *et al.*, octobre 2002] illustre parfaitement cette approche en combinant WordNet et une ontologie du domaine. Les auteurs ont montré dans une expérimentation sur le domaine de la technologie que leur système est deux fois plus précis que Google sur les mêmes requêtes.

Pour augmenter la précision, d'autres travaux préconisent l'amélioration de la qualité des documents contenus dans la collection sur laquelle vont porter les requêtes. Deux améliorations possibles ont été proposées : n'inclure dans la collection que les pages populaires (les mieux classées) ou restreindre le domaine des documents à un thème fixé. Dans le premier cas de figure, *Junghoo Cho et al* [Cho *et al.*, 1998] ont montré que rapatrier prioritairement les pages ayant un PageRank élevé augmentait considérablement la qualité des documents de la collection. De ce fait, les réponses à une requête posée sur une telle collection sont de qualité (PageRank élevé) et ont de fortes chances de correspondre aux souhaits de l'utilisateur. Dans le deuxième cas de figure (fixer un thème précis), deux techniques majeures ont été étudiées : l'exploration ciblée du web (*focused crawling* [Diligenti *et al.*, 2000, Rennie et McCallum, 1999]) et l'exploration intelligente (*intelligent crawling* [Aggarwal *et al.*, 2001]).

L'exploration ciblée du web consiste à restreindre l'espace d'exploration du web pour ne rapatrier que les documents jugés en rapport avec la thématique fixée. Pour cela, une stratégie sélective d'exploration du web, du type « le meilleur d'abord », est mise en place pour suivre en priorité les liens jugés prometteurs (menant rapidement vers beaucoup de documents pertinents). Cette stratégie repose sur une fonction discriminante f_{lien} qui à partir des mots apparaissant dans le voisinage d'un lien estime l'intérêt de le suivre. Cette fonction f_{lien} est construite à la suite d'un processus d'apprentissage à partir d'un échantillon du graphe du web représentatif des documents de la thématique fixée. Par exemple, dans [Rennie et McCallum, 1999], les auteurs se sont intéressés à la construction d'un moteur de recherche *CORA* [whizbang, 2001], spécialisé dans les papiers scientifiques en ligne. Ils ont élaboré une fonction d'estimation de l'intérêt d'un lien en combinant deux techniques d'apprentissage (apprentissage par renforcement et classifieur bayésien) sur un échantillon du graphe du web. Dans ce même domaine, les travaux de *Diligenti & al.* [Diligenti *et al.*, 2000] ont servi de base pour le développement du moteur de recherche *CiteSeer* [CiteSeer, 2003]. Leur approche consiste à utiliser un classifieur bayésien pour apprendre à estimer la distance qui sépare le document courant d'un ou plusieurs éventuels documents pertinents. Leur stratégie d'exploration privilégie les liens sortants d'un document jugé proche de documents pertinents. Le focused crawling est une alternative bien adaptée à la problématique de la recherche ciblée de documents sur le web mais reste assez lourde à mettre en œuvre. En effet, les techniques d'apprentissage, sur lesquelles l'exploration sélective repose, nécessitent un échantillon contenant suffisammment de documents représentatifs du domaine pour atteindre des performances acceptables. Ce type d'échantillons n'est pas toujours évident à construire et nécessite une lourde intervention humaine dans le processus d'étiquetage manuel des documents pertinents.

L'exploration intelligente (*Intelligent Crawling*) a été proposée par *Charu C. Aggarwal & al* dans leurs travaux [Aggarwal *et al.*, 2001]. Cette approche ne nécessite aucun apprentissage au préalable. Partant de certains points de départ, un robot « intelligent » apprend au fur et à mesure de son exploration à privilégier les liens prometteurs. Cet apprentissage progressif repose sur une fonction discriminante qui évalue chaque page rapatriée. Par exemple, la vérification de la présence d'une liste de mots prédéfinis peut servir de fonction discriminante. Ainsi, à chaque nouvelle page rapatriée et évaluée par la fonction discriminante, une combinaison de mesures statistiques est calculée pour toutes les pages candidates (liens non encore suivis). Seule la page candidate totalisant un score de combinaison élevé est rapatriée. Les mesures statistiques utilisées dans cette approche traduisent en valeurs numériques plusieurs aspects et caractéristiques du graphe du web exploré jusque là. Par exemple, une des mesures que les auteurs ont utilisé consiste à traduire en un score la probabilité pour qu'une page candidate, pointée par un certain nombre de pages pertinentes (évaluées à vrai par la fonction discriminante), soit elle aussi une page pertinente et donc intéressante à rapatrier. Cette mesure évolue à chaque étape de l'exploration et dépend du nombre de pages pertinentes rapatriées au total. L'exploration intelligente possède un avantage certain sur l'exploration ciblée car elle réduit les coûts d'apprentissage initiaux (pas de graphe d'apprentissage ni d'étiquetage humain a priori). Cependant, l'efficacité d'une telle approche dépend essentiellement de la fiabilité de la fonction discriminante à discriminer entre pages pertinentes et pages non pertinentes.

En résumé, la clé de la recherche ciblée de documents sur le web réside dans la capacité de caractériser clairement et facilement les pages d'intérêt pour pouvoir, par la suite, les distinguer sans ambiguïté des pages non pertinentes et à moindre coût. Dans ce sens, nous proposons un langage de requêtes déclaratif qui permet de caractériser les pages souhaitées d'une manière plus fine que les moteurs de recherche actuels.

3 Notre langage de requêtes

Nous avons défini un langage de requêtes de type attribut-valeur qui permet à un utilisateur de combiner, à l'aide d'opérateurs logiques, plusieurs critères pour caractériser les pages qui l'intéressent. Chaque critère spécifié dans une requête permet de cibler la recherche de ses valeurs (mots-clés) sur une partie bien déterminée de la structure d'une page (exemple : son titre, le voisinage de ses liens sortants ...) ou de caractériser une propriété particulière d'une page (exemple : son url, type mime...). En utilisant les opérateurs logiques de conjonction et de disjonction, il est possible de combiner les critères précédents de manière à cibler à la fois le type de la page (html, ps, pdf, jpg...) avec : certaines propriétés de l'url d'une page, des caractéristiques de certaines de ses parties clés (titre, voisinage de liens sortants ou entrants ...). Toute requête de notre langage est construite à partir de la combinaison logique de requêtes atomiques. Une requête atomique est une requête mots-clés de type « attribut = valeurs » où les attributs sont fixés. Les différentes requêtes atomiques que nous considérons sont :

• $\mathbf{page\_title\ =\ val_1, \ldots, val_n}$ **:** où chaque val_i est une chaîne de caractères pouvant contenir plusieurs mots séparés par le caractère blanc. Une telle requête est évaluée à $vrai$ sur une page p si et seulement si tous les mots apparaissant dans au moins une des chaînes de caractères val_i sont contenus dans le titre de cette page p dans l'ordre dans lequel ils sont déclarés dans val_i. Le titre d'une page, lorsqu'il existe, est repéré par le contenu des balises <title> ou <h1> ou <meta name="title">.

Exemple : Soit la requête atomique q_a
« ***page_title = cours java, introduction à java*** »

titre de la page	évaluation de q_a
cours de licence en java	$vrai$
java cours introductif	$faux$

• $\mathbf{mime\ =\ type_1, \ldots, type_n}$ **:** Les types $type_i$ doivent être conformes au standard défini pour les types mime de documents sur le web du langage html. La requête est évaluée à $vrai$ sur une page p si le type mime de p correspond à l'un des types spécifiés dans la liste des $type_i$.

Exemple : Soit q_a :
mime = application/postscript, application/pdf

type de la page	évaluation de q_a
document pdf	$vrai$
document postscript	$vrai$
document html	$faux$

• $\mathbf{title\_incoming\_page\ =\ val_1, \ldots, val_n}$ **:** Cette requête est évaluée à $vrai$ sur une page p si et seulement s'il existe une page p' ayant un lien pointant vers la page p et telle que la requête $page\_title\ =\ val_1, \ldots, val_n$ soit évaluée à $vrai$ sur p'.

• $\mathbf{url\ =\ val_1, \ldots, val_n}$ **:** Cette requête est évaluée à $vrai$ sur une page p si et seulement si tous les mots d'un des val_i apparaissent (en respectant leur ordre dans ce même val_i) dans les tokens de l'url de p.

Exemple : Soit q_a :
url = univ cours java

url de la page	évaluation de q_a
`http://www.infres.enst.fr/~charon/coursJava/`	$faux$
`http://www.univ-reunion.fr/~courdier/cours/java`	$vrai$

• $\mathbf{incoming\_links} \ = \ \mathbf{val_1}, \ldots, \mathbf{val_n}$ **:** Cette requête permet de cibler les mots apparaissant dans le voisinage des liens entrants d'une page. Nous avons défini le voisinage d'un lien comme étant : les mots de son ancre, les tokens de l'url à laquelle il fait référence ainsi que les 10 mots précédant et suivant le lien (avant le tag <a href> et après le tag fermant < /a>. Cette requête est évaluée à $vrai$ sur une page p si et seulement s'il existe un lien l pointant vers p tel qu'il existe val_i pour qui tous ses mots soient présents (dans leur ordre dans val_i) dans le voisinage de l.
• $\mathbf{outgoing\_links} \ = \ \mathbf{val_1}, \ldots, \mathbf{val_n}$ **:** Cette requête permet de cibler les mots apparaissant dans le voisinage des liens sortants d'une page. Cette requête est évaluée de la même manière que la requête précédente sauf qu'elle porte sur les liens même d'une page.
• $\mathbf{url\_length} \ = \ \mathbf{number\_restriction}$ **:**
où $number\_restriction$ est une restriction de cardinalité de la forme atleast[n] ou atmost[n]. Cette requête permet de contraindre la longueur d'une url. Nous avons défini la longueur d'une url comme étant le nombre de '/' qu'elle contient.

Exemple : ***url_length = atmost[2]*** nous a été utile pour caractériser la notion de page d'accueil.

url de la page	évaluation de q_a
`http://www.gofast.com`	$vrai$
`http://www.yahoo.fr/tourisme`	$faux$

Dans la suite, nous noterons $\mathcal{G}(\mathcal{P}, \mathcal{L})$ le graphe des pages web sur lequel nous évaluons les requêtes de notre langage où $\mathcal{P}$ est l'ensemble des nœuds (pages webs) du graphe $\mathcal{G}$ et $\mathcal{L}$ l'ensemble des arcs (liens hypertexte) reliant les pages de l'ensemble $\mathcal{P}$ entre-elles.
La sémantique $\mathcal{S}_{q_a}$ associée à une requête atomique q_a est définie par l'ensemble des réponses issues de l'évaluation de q_a sur les ensembles $\mathcal{P}$ et $\mathcal{L}$:

$$\mathcal{S}_{q_a} \ = \ \{url(p) \ / \ (p \ \in \ \mathcal{P}) \wedge (q_a \text{ est évaluée à } vrai \text{ sur } p)\}$$

Pour l'évaluation des requêtes atomiques, nous avons implanté en java un programme qui, pour chaque page de $\mathcal{P}$, apparie le contenu de la section ciblée par la requête atomique q_a aux expressions régulières générées à partir des valeurs contenues dans q_a et conformément à la syntaxe unix/java/perl. La génération d'expressions régulières est obtenue en remplaçant les caractères blancs (séparant les mots d'une valeur val_i) par «.*» (ce qui signifie que les mots sont recherchés dans leur ordre dans val_i indépendamment des caractères qui puissent se trouver entre eux). Il est possible d'ajouter, aux attributs fixés de notre langage (voir section 3), d'autres attributs dès lors qu'ils soient évaluables sur le graphe $\mathcal{G}$.

Une requête conjonctive dans notre langage est une conjonction de requêtes atomiques sans répétition d'attributs et ayant un et un seul attribut relatif au type mime. La sémantique d'une conjonction de requêtes atomiques $q \ = \ q_a^1 \ \wedge \ \ldots \ \wedge \ q_a^n$ est définie par : $\mathcal{S}_q \ = \ \bigcap_{i=1}^{n} \ \mathcal{S}_{q_a^i}$
Une requête disjonctive dans notre langage est une disjonction de conjonctions de requêtes atomiques. La sémantique de $Q \ = \ q_1 \ \vee \ \ldots \ \vee \ q_n$ est définie par : $\mathcal{S}_Q \ = \ \bigcup_{i=1}^{n} \ \mathcal{S}_{q_i}$
Exemple : Soit la requête suivante pour rechercher des cours java au format pdf, ps ou html

(mime=application/pdf,application/postscript) $\wedge$ ***(incoming_links=cours java,introduction java)*** $\wedge$
(url=univ java,enseignement java) $\bigvee$ ***(mime=text/html)*** $\wedge$ ***(url=univ cours java)*** $\wedge$
(outgoing_links=sommaire,intro) $\wedge$ ***(page_title=introduction java)*** $\wedge$ ***(incoming_links=cours java)***

4 Évaluation expérimentale de notre langage

Afin d'évaluer notre langage de requêtes, nous avons effectué différents tests en suivant un protocole expérimental spécifiant l'échantillon de requêtes à tester, le corpus de pages web pour l'évaluation et les mesures de qualité du langage.

4.1 L'échantillon de requêtes testées

Les requêtes, que nous avons testées, portent sur la recherche de documents spécifiques dans 3 domaines différents : des documents de cours en informatique en français, des pages d'accueil de services touristiques en français et des photos de formule 1. En nous mettant à la place d'un utilisateur expérimenté recherchant de tels documents, nous avons manuellement élaboré 13 requêtes différentes réparties comme suit : 4 requêtes pour les documents de cours en informatique, 5 requêtes pour les documents de tourisme et 4 requêtes pour les photos de formule 1. Dans un premier temps et pour chaque domaine, les différentes requêtes testées ont été obtenues par l'élaboration d'une requête que nous appelons : « *requête initiale* ». Dans cette requête, le rôle de l'utilisateur expérimenté a consisté à choisir : les attributs à cibler, les valeurs mots-clés à renseigner pour chaque attribut retenu et la combinaison logique à former sous forme de disjonctions de conjonctions d'attributs. Les requêtes initiales des trois domaines sont données dans le tableau Tab.1. Les requêtes initiales exploitent la richesse de notre langage de requêtes pour spécifier le plus précisément possible les documents à rechercher dans chaque domaine. L'élaboration de ces trois requêtes permet de mettre en évidence le pouvoir d'expression de notre langage. Par la suite, les autres requêtes du domaine sont générées par variation de la requête initiale et sont identifiées par « *variation* k » de la requête initiale. Dans une variation k d'une requête initiale $\mathcal{Q}$, k requêtes atomiques de toutes les conjonctions de $\mathcal{Q}$ sont relaxées pour obtenir une autre requête moins restrictive que la requête initiale.
Soit $q_1 \ \wedge \ \ldots \ \wedge \ q_n$ une conjonction quelconque d'une requête initiale $\mathcal{Q}$. Relaxer k requêtes atomiques de cette conjonction où $n \ > \ k$ consiste à remplacer cette conjonction par la disjonction des $\mathcal{C}_n^k$ conjonctions possibles suivantes : $q_{i_1} \ \wedge \ldots \wedge \ q_{i_{n-k}}$. Pour des raisons de cohérence sémantique, la requête atomique relative au type mime d'une page n'est pas relaxable. Elle doit nécessairement être combinée (par conjonction) avec au moins une autre requête atomique. En conséquence, il ne peut y avoir de variations k pour $k \ \geq \ n-2$.
Par ailleurs, et de par leur construction, chaque variation k d'une requête $\mathcal{Q}$ est moins restrictive et donc moins précise que la requête initiale $\mathcal{Q}$. De plus, une variation k de $\mathcal{Q}$ est plus restrictive (donc plus précise) qu'une variation $k+1$ de $\mathcal{Q}$. Les différentes variations des trois requêtes initiales de chaque domaine d'expérimentation ont été définies et évaluées dans le but de mesurer, d'une part, l'intérêt des combinaisons logiques (plus précisément la pertinence des conjonctions de requêtes atomiques) et de pouvoir, d'autre part, mettre en évidence le gain en précision. Ce gain est montré dans le tableau Tab.3.

4.2 Corpus des pages évaluées

Pour évaluer nos requêtes, nous avons été confrontés au problème du choix du graphe de pages web à construire. Pour des raisons de limitation de capacité de stockage et de temps d'exploration du web, nous ne pouvons pas évaluer nos requêtes sur tout le web (ou du moins sur un index équivalent en taille à celui de Google). Nous avons alors étudié d'autres alternatives moins coûteuses et mieux adaptées à nos moyens matériels. Nous avons identifié deux possibilités différentes : construire un graphe de manière aléatoire ou construire 3 graphes

Pages d'accueil de services touristiques
(URL = voyage,tourisme,sejour,reservation billet . . .) $\wedge$ (page_title = agence tourisme,tour operateur,office tourisme,compagnie aerienne,vol charter . . .) $\wedge$ (Incoming_Links = agence tourisme, tour operateur,achat reservation billets,vol regulier,voyage discount. . .) $\wedge$(Outgoing_Links = reservez, vol,sejour,contact . . .) $\wedge$ (MIME = text/html) $\wedge$ (url_length = atmost[2])
Documents de cours en informatique
(url = cours cpp,cours slide. . .)$\wedge$ (page_title = cours informatique,cours algo. . .) $\wedge$ (Incoming_Links = cours algo,cours info . . .) $\wedge$ (Outgoing_Links = introduction,sommaire. . .)$\wedge$ (MIME = text/html) $\bigvee$ (Incoming_Links = cours reseaux,cours 'ia',support cours informatique, documentation cours) $\wedge$ (URL = cours ia,univ $\sim$ cours,info cours . . .) $\wedge$ (MIME = application/postscript,application/pdf . . .) $\bigvee$ (title_Incoming_Page = cours informatique,documentation cours,notes de cours,cours algo . . .) $\wedge$ (MIME = application/ppt,text/htm,application/pdf . . .) $\wedge$ (URL = cours 'bd',$\sim$ sld . . .)
Photos de formule 1
(page_title = photo gallery 'f1',schumacher picture,montoya picture . . .) $\wedge$ (URL = photo 'f1' coulthard picture,villeneuve picture . . .) $\wedge$ (Incoming_Links = photo formule 1,image formule 1, grand prix photo . . .) $\wedge$ (Outgoing_Links = suivant,precedent, previous. . .) $\wedge$ (MIME = text/htm) $\bigvee$ (URL = f1 photo,image f1,grand prix photo,gallerie f1 . . .) $\wedge$ (Incoming_Links = formula one pic, f1 picture,formula one gallery . . .) $\wedge$ (MIME = image) $\bigvee$ (URL = photo ferrari f1,photo mac laren f1,ralf picture . . .) $\wedge$ (title_Incoming_Page = formula one gallery,photo ferrari 'f1' . . .) $\wedge$ (MIME = image,text/htm)

TAB. 1 – *Les 3 requêtes initiales d'expérimentation*

thématiques en rapport avec les 3 domaines fixés dans la section précédente. La première possibilité est incontestablement la plus triviale à mettre en œuvre. Cependant, elle présente un inconvénient majeur : aucune garantie d'avoir des pages pertinentes dans le lot aléatoire. Ceci peut engendrer un biais dans les résultats des évaluations. Nous avons donc retenu la deuxième possibilité : construire 3 graphes thématiques.

Pour construire nos 3 graphes thématiques, nous avons utilisé Google pour recueillir, dans un premier temps, des urls en rapport avec les 3 thématiques fixées. Certaines de ces urls sont pertinentes d'autres ne le sont pas. Toutes ces urls sont rapatriées et stockées en local formant 3 graphes de pages (un graphe par domaine). En moyenne, les ensembles des sommets de ces graphes totalisent une taille d'environ 2000 sommets (pages) (voir le tableau Tab.2). Par la suite, notre robot-explorateur a exploré le web pour étendre les 3 graphes précédents aux pages référencées par les liens sortants des pages données par Google. Nous avons choisi de paramétrer notre robot de sorte qu'il ne suive que les liens sortant des pages les plus pertinentes. La pertinence de ces pages est déterminée par le classement des urls de ces pages dans la liste des réponses Google. À la fin de cette première étape, nous obtenons un niveau supplémentaire pour chaque graphe thématique à construire. Ce niveau correspond à l'extension des meilleures réponses de Google d'un pas. Le processus d'extension des graphes est répété p fois donnant un graphe final par domaine de profondeur p où chaque niveau est construit comme suit :

– le niveau zéro contient les pages dont les urls sont fournies par Google en réponses à nos requêtes thématiques ;

– le niveau 1 est constitué de l'extension des pages les plus pertinentes du niveau 0 ;

– Les niveaux $i,\ 1 < i < p$ sont construits à partir des pages issues du crawling semi-exhaustif du web à partir des liens sortants du niveau $i - 1$. Afin de satisfaire les contraintes imposées par nos moyens matériels, nous avons défini un facteur de branchement b variable pour réduire le nombre de liens à suivre par page.

Les choix, que nous avons retenus pour la construction de nos graphes thématiques, nous garantissent la présence d'un nombre suffisant de pages pertinentes pour valider les mesures de qualité (voir section 4.3) des évaluations de nos requêtes. La justification de ces choix repose sur deux hypothèses. D'une part, nous considérons Google comme suffisamment puissant et fiable pour fournir des urls de qualité. Cette qualité est particulièrement confirmée lorsqu'il s'agit des pages bien classées dans les réponses [Brin et Page, 1998]. D'autre part, l'extension des meilleures réponses de Google par crawling pour inclure plus de pages pertinentes s'appuie sur les travaux montrant que le web est thématiquement connexe [Kleinberg, 1999, Brin et Page, 1998, Diligenti *et al.*, 2000]. En effet, une page pertinente pointe souvent sur d'autres pages pertinentes traitant du même thème. Suivre les liens d'une telle page augmente la chance d'inclure des pages pertinentes dans le graphe thématique.

Soit $\mathcal{Q}$ la requête initiale d'un des trois domaines fixés. Le graphe de tests $\mathcal{G}(\mathcal{P}, \mathcal{L})$ associé à ce domaine est construit suivant les étapes décrites ci-dessous :

– Au départ, un ensemble de requêtes Google est constitué à partir des requêtes atomiques contenues dans la requête $\mathcal{Q}$. Seules les requêtes atomiques exécutables par Google avancé sont retenues et sont : $page\_title$, url, $incoming\_links$, $outgoing\_links$. Pour chaque occurence de l'une de ces dernières requêtes dans $\mathcal{Q}$, nous soumettons toutes ses valeurs à Google avancé. En réponses, nous récupérons au maximum 100 urls (limite de l'API Google) par valeur soumise.

  Exemple : Soit $\mathcal{Q}$ la requête suivante : ***(page_title = cours java , cours c++)*** $\vee$ ***(url = univ java , cours cpp)***. La syntaxe de la requête Google avancé envoyée est : ***(allintitle: cours java) OR (allintitle: cours c++) OR (allinurl: univ java) OR (allinurl: cours cpp)***.

  Nous avons retenu toutes les urls fournies et leur classement (pagerank) associé obtenant ainsi un ensemble initial d'urls. Les urls de cet ensemble sont rapatriées et sauvegardées dans une base de données MySql suivant un schéma que nous avons défini et qui nous permet d'évaluer les attributs de nos requêtes. Les informations d'une page que nous avons gardées dans notre base sont : son titre, son url, son type mime, l'ancre et le voisinage de ses liens sortants et lorsque cela est possible l'ancre et le voisinage des liens qui pointent vers cette pages. Les tuples de notre base de données MySql constituent une sous-partie de notre ensemble de tests $\mathcal{P}$.

– Pour construire le graphe, nous avons choisi d'étendre les deux meilleures urls par valeur soumise. Pour cela, nous avons conçu un robot explorateur en java qui parcours le web suivant la stratégie « profondeur d'abord ». Le robot admet en entrée deux paramètres : les urls de départ (dans notre cas les deux meilleures urls par valeur soumise), la profondeur maximale à atteindre (10 dans nos expérimentations). Pour limiter l'espace d'exploration, le facteur de branchement que nous avons défini est le suivant : facteur de branchement par page $b = \frac{1}{i+1}\%$. Ce facteur nous a permis de ne retenir que $b\%$ liens, choisis aléatoirement parmi tous les liens d'une page d'un niveau $i, 1 < i < p$.

– La taille du graphe construit pour chaque domaine est donné dans le tableau Tab.2.

requête	# réponses Google	# pages $\|\mathcal{P}\|$	# liens suivis $\|\mathcal{L}\|$
cours informatiques en ligne	2.404	271.850	4.347.930
annuaire de documents de tourisme	2.027	239.817	5.916.248
photos de formule 1	1.599	149.634	8.990.615

TAB. 2 – *Taille du graphe pour chaque requête test*

4.3 Mesures de qualité

Nous avons retenu deux mesures pour évaluer la qualité des réponses obtenues : La précision et la couverture. La précision est la proportion de pages réellement pertinentes parmi les pages réponses de la requête (évaluées à $vrai$). La couverture est le taux de pages réellement pertinentes des réponses de la requête parmi les pages pertinentes du graphe $\mathcal{P}$.

Soient $\mathcal{P}$ l'ensemble de pages de tests, $\mathcal{Q}$ la requête à évaluer et $\mathcal{S}$ l'ensemble des réponses issues de l'évaluation de $\mathcal{Q}$ sur $\mathcal{P}$. La modalité de pertinence nous permet de diviser chacun des ensembles précédents en deux sous-ensembles de pages réellement pertinentes ($\mathcal{P}^p$) et pages non-pertinentes ($\mathcal{P}^{np}$) : $\mathcal{P} = \mathcal{P}^p \cup \mathcal{P}^{np}$, $\mathcal{S} = \mathcal{S}^p \cup \mathcal{S}^{np}$. La précision et la couverture se définissent alors par :

$$\text{précision} = \frac{|\mathcal{S}^p|}{|\mathcal{S}|} \qquad \text{couverture} = \frac{|\mathcal{S}^p|}{|\mathcal{P}^p|}$$

La taille du graphe de tests (voir Tab.2) et des réponses retournées lors des évaluations étant importantes, nous avons eu recours à des techniques d'échantillonnage pour estimer la précision et la couverture. Nous avons constitué, à chaque évaluation, un échantillon aléatoire de 100 pages et nous l'avons manuellement étiqueté en pertinent ou pas (estimation de $|\mathcal{S}^p|$). Nous avons fait de même pour estimer la proportion des pages pertinentes contenues dans $\mathcal{P}$ ($|\mathcal{P}^p|$)
Nous voulons pouvoir estimer le gain de précision d'une requête de notre langage face à Google (Google classique et Google avancé). La différence du pouvoir d'expression entre notre langage de requêtess et celui de Google (classique ou avancé) fait que nous ne pouvons pas écrire une requête exécutable par Google qui soit équivalente à une requête de notre langage. En effet, nos requêtes initiales de tests et leurs variations sont des combinaisons logiques de requêtes atomiques. Or, Google, dans sa version classique, ne permet pas de cibler les parties d'une page contrairement à une requête WeQueL. Dans sa version avancé, Google permet seulement de cibler des parties particulières d'une page sans pouvoir les combiner. En conséquence et pour chaque domaine, nous avons défini plusieurs requêtes Google comparables à nos requêtes de tests construites comme suit :

– une requête classique correspondant à l'union de toutes les valeurs mots-clés apparaissant dans les différentes requêtes atomiques de la requête initiale ;

– autant de requêtes Google avancé que de requêtes atomiques exécutables par Google avancé dans la requête initiale $\mathcal{Q}$.

De cette manière, les comparaisons de nos requêtes avec Google mettent en avant deux qualités essentielles de notre langage : l'efficacité des requêtes ciblées sur des parties particulières de pages (comparaison avec Google classique) et le gain en précision par combinaison logique de requêtes atomiques (comparaison avec Google avancé et avec les différentes variations). La couverture de Google est calculée à partir du taux de pages pertinentes que couvrent les réponses Google sur notre graphe de tests $\mathcal{P}$. De même que pour les requêtes de notre langage, nous estimons par échantillonnage la précision et la couverture des résultats des requêtes Google (avancé et classique). Les résultats de ces estimations sont données dans le tableau Tab.3. Dans ces tableaux, nous ne faisons apparaître dans la ligne « Google avancé » que la meilleure précision et la meilleure couverture issues des évaluations des requêtes « Google avancé » testées.

5 Résultats expérimentaux

Nous avons été extrêmement restrictifs dans nos étiquetages manuels. Par exemple, les documents qui en soi ne sont pas pertinents mais font référence à des documents pertinents sont étiquetés comme non pertinents. Seuls les documents ayant un contenu en rapport direct avec les thématiques fixées sont étiquetés à pertinent. Ceci explique en partie les faibles scores de précision de Google et de certaines de nos variations. Les résultats obtenus sur les trois thèmes fixés sont résumés dans le tableau Tab.3. Il apparaît très clairement des deux premières lignes

	cours informatiques		**annuaire de tourisme**		**photos de formule 1**	
requête	***précision***	***couverture***	***précision***	***couverture***	***précision***	***couverture***
Google classique	9,00%	6,68%	10,00%	7,29%	9,00%	22,99%
Google avancé	26,00%	5,82%	21,00%	4,14%	32,00%	9,47%
requête initiale	71,00%	11,16%	100,00%	2,30%	65,53%	9,44%
variation 1	56,00%	56,57%	29,29%	10,56%	41,00%	36,11%
variation 2	65,00%	63,31%	8,00%	21,74%	36,00%	35,99%
variation 3	10,00%	100%	7,00%	61,56%	8,00%	93,01%
variation 4	#	#	2,00%	100%	#	#

TAB. 3 – *Résultats expérimentaux*

de chaque tableau (résultats de Google) qu'une requête ciblée est plus précise qu'une requête classique portant sur tout le contenu d'une page web. Pour les requêtes ciblée Google, nous avons évalué à chaque fois trois possibilités qu'offrait Google avancé : le titre, l'url et puis les liens entrants. Nous n'avons retenu que la meilleure évaluation.Les requêtes sur les liens entrants ont donné les résultats les plus bas. Ces derniers résultats sont pratiquement équivalents aux résultats des requêtes classiques. Par contre, les requêtes Google ciblées sur le titre ou sur l'url ont atteint des précisions équivalentes, sensiblement plus importantes que les précisions des requêtes classiques.

En analysant les résultats des lignes Google avancé et nos requêtes initiales, nous remarquons qu'une requête de notre langage est en moyenne trois fois plus précise que Google avancé (avec les mêmes mots-clés, voir protocole expérimental dans la section 4.3). Ceci montre expérimentalement l'intérêt et le gain en précision d'une requête combinant plusieurs critères différents. Nous pouvons ainsi conclure qu'une requête de notre langage est plus expressive qu'une requête mots-clés et permet de mieux cerner les documents à rechercher.

En comparant les résultats de l'évaluation de chacune de nos requêtes aux résultats des variations qui leur correspondent, nous remarquons que la précision décroît au fur et à mesure

que l'on relâche les contraintes d'évaluation (c'est-à-dire moins de critères par conjonction). Ceci met en évidence l'apport en précision des combinaisons par opérateurs logiques (plus précisément la conjonction de plusieurs critères à la fois). En effet, une conjonction faisant intervenir plusieurs critères à la fois permet une spécification plus riche et plus fine des documents à rechercher qu'une conjonction qui fait intervenir moins de critères.
Cependant, la couverture des requêtes de notre langage est assez faible au vu des documents pertinents dans notre ensemble de tests. Ceci s'explique par notre méthode d'évaluation qui consiste à n'apparier que les mots fixés dans les requêtes. Il est évident que l'évaluation de nos requêtes passe à côté des documents pertinents ne contenant pas les termes de la requête mais des synonymes équivalents par exemple. De plus, le concepteur de la requête, aussi expérimenté qu'il soit, ne peut renseigner de manière exhaustive sa requête. Une possible amélioration consiste donc à enrichir la requête initiale par les synonymes des mots des valeurs de chaque requête atomique (utiliser WordNet ou une ontologie). Il est également possible d'avoir recours à des techniques d'apprentissage pour déduire de nouveaux mots pertinents ne figurant pas dans la requête initiale.

6 Conclusion et perspectives

Dans cet article, nous avons présenté un langage de requêtes pour cibler de manière plus efficace les pages pertinentes à rechercher. Nous avons illustré son prouvoir d'expression par des exemples de requêtes pour rechercher trois types de documents (cours informatique, documents de tourisme, photos de f1). Nous avons, par ailleurs, montré expérimentalement que notre langage est significativement plus précis que Google sur des requêtes comparables.
Notre langage est en cours d'utilisation dans le projet « eDot » [eDot, 2002] comme outil d'aide à la création d'entrepôts de données thématiques. Le projet « eDot » (Entrepôts de Données Ouverts sur la Toile) consiste à développer un entrepôt de données alimenté à partir du web et contenant les documents traitant du risque alimentaire. Dans ce projet, nous utilisons la puissance d'expressivité et la précision de notre langage pour définir et spécifier les besoins de l'entrepôt en terme de pages web en élaborant une requête caractéristique. Contrairement aux requêtes thématiques de cet article, la requête caractéristique d'« eDot » a été élaborée de manière semi-automatique en s'appuyant sur une ontologie du domaine et un ensemble de pages exemples fournies par un expert.
Nous étudions également dans le projet « eDot » le comportement de notre langage comme outil de filtrage de pages web. Les pages web à filtrer sont rapatriées par le crawler de Xyleme [Xyleme, 2002]. Ce crawler a été paramétré de manière à ce qu'il ne rapatrie que les pages contenant les mots clés de l'ontologie utilisée (Sym Previous) et cela indépendamment de leur localisation dans la page. Ceci constitue un premier filtrage du web. La requête caractéristique aura donc pour but de cibler les mots-clés de l'ontologie sur certaines parties d'une page et d'estimer par la suite la précision des résultats obtenus. En fonction des résultats que nous obtiendrons, nous pourrons avoir recours à un crawling intelligent pour rapatrier directement du web les pages jugées pertinentes et augmenter par conséquent la couverture de l'entrepôt.

Références

[Aggarwal *et al.*, 2001] Charu C. Aggarwal, Fatima Al-Garawi, et Philip S. Yu. Intelligent crawling on the world wide web with arbitrary predicates. In *World Wide Web*, pages 96–

105, 2001.

[Brin et Page, 1998] Sergey Brin et Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[Cho *et al.*, 1998] Junghoo Cho, Hector García-Molina, et Lawrence Page. Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1–7):161–172, 1998.

[CiteSeer, 2003] CiteSeer. http://citeseer.nj.nec.com/cs, 2003.

[Diligenti *et al.*, 2000] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C. Lee Giles, et Marco Gori. Focused crawling using context graphs. In *26th International Conference on Very Large Databases, VLDB 2000*, pages 527–534, Cairo, Egypt, 10–14 September 2000.

[eDot, 2002] eDot. Entrepôts de données ouverts sur la toile : http://www-rocq.inria.fr/verso/gemo/projects/edot.pdf, 2002.

[Felbaum, 1998] C. Felbaum, editor. *WordNet: an electronic lexical database*. Boston: MIT Press, 1998.

[Google, 2003] Google. http://www.google.com, 2003.

[Halkidi *et al.*, octobre 2002] Maria Halkidi, Benjamin Nguyen, Iraklis Varlamis, et Michalis Vazirgiannis. Organising web documents into thematic subsets using an ontology (THESUS). In *Actes électroniques des Journees Web Semantique*, Paris, octobre 2002.

[Kleinberg, 1999] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[Rennie et McCallum, 1999] Jason Rennie et Andrew Kachites McCallum. Using reinforcement learning to spider the web efficiently. In *Proc. 16th International Conf. on Machine Learning*, pages 335–343. Morgan Kaufmann, San Francisco, CA, 1999.

[Voorhees, 1998] EM. Voorhees. *WordNet: An Electronic Lexical Database and some of its Applications*, chapter 12: Using WordNet for Text Retrieval. MIT Press, Christiane Fell-Baum editor, 1998.

[whizbang, 2001] whizbang. Cora version 2.0 : Computer science research paper search engine, 2001. http://cora.whizbang.com.

[Xyleme, 2002] Xyleme. http://www.xyleme.com/, 2002.

Summary

Keyword-based web query languages suffer from a lack of precision when searching for a precise kind of documents. Indeed, some documents cannot be simply characterized by a list of keywords (e.g. on-line java courses or pictures of formula one). We propose a multi-criteria query langage for a better characterization of documents. The aim is to increase the precision of document retrieval on the web. In our experiments, we show the gain in accuracy in web document searching using our language.