TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

Moteurs de recherche

La *recherche d'information* est un domaine de recherche fort de plus de cinquante années de travaux. Tourné à ses débuts vers les bibliothécaires et les professionnels de l'information, le domaine a connu un bouleversement majeur dans les années 90 : l'arrivée d'Internet et plus précisément du *World Wide Web* (ou, plus simplement, *Web*), qui deviendra par la suite l'application fédératrice des recherches de ce domaine. Le *Web* a permis à des millions d'utilisateurs de créer, publier et diffuser du contenu au niveau international. Il est aujourd'hui, sans nul doute, le plus important répertoire de connaissance de l'histoire de l'humanité. Le volume de documents disponibles est en augmentation constante [1], atteignant, selon de récentes estimations [2], le chiffre astronomique d'un millier de milliards de documents. Face à cette masse de connaissances, il est rapidement devenu indispensable d'employer des outils permettant de trouver une réponse pertinente à nos *besoins d'informations* quotidiens : les *moteurs de recherche*.

Le terrain fertile du *Web* a permis l'éclosion de plusieurs moteurs de recherche grand public : Excite (1994), AltaVista (1995), Infoseek (1995), Inktomi (1996) et Google (1997/98). Google domine aujourd'hui encore ce domaine avec plus de 85 % de part de marché dans le monde [3]. Seules l'Asie et l'Afrique de l'Ouest (avec Yahoo !), la Russie (avec Yandex) ou la Chine (avec Baidu) montrent une forme d'opposition au monopole de Google (Figure 1 d'après multilingualwebmarketing.com).

Chaque année, de nouveaux moteurs de recherche tels que Powerset, Cuil, Wolfram Alpha, DuckDuckGo ou Blekko tentent de détrôner le géant du Web, mais aboutissent à un résultat mitigé. Google reste le site internet le plus visité au monde [4], et possède même un verbe outre-Atlantique :

> *goo·gle : to use the Google search engine to obtain information about a subject on the World Wide Web (Dictionnaire Merriam-Webster).*

Toutefois, l'approche généraliste des moteurs de recherche laisse la porte ouverte aux moteurs de recherche spécialisés, aussi nommés moteurs de recherche verticaux, qui sont l'objet de cette thèse. D'une part, les moteurs de recherche spécialisés visent à améliorer la pertinence des résultats, à

1. Un ensemble d'études à ce sujet sont présentées dans (Liu, 2009, chap. 1, p. 3).
2. http://googleblog.blogspot.fr/2008/07/we-knew-web-was-big.html
3. Chiffres datant d'octobre 2012 issus des sites Alexa et StatsCounter - GlobalStats.
4. Chiffres Alexa.com, novembre 2012.

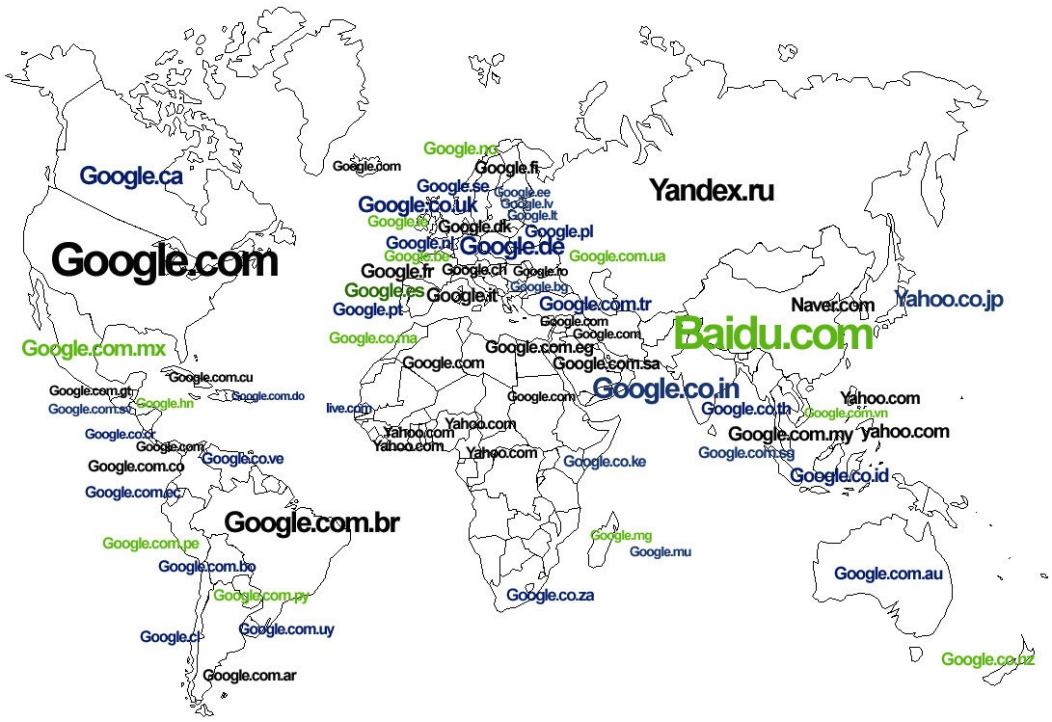


FIGURE 1 : Carte du monde des moteurs de recherche les plus utilisés par pays, basée sur les données du site Alexa.com en août 2010.

réaliser des recherches plus fines et à diminuer le temps de recherche au travers de données et d'outils dédiés. D'autre part, ils offrent un changement de paradigme de recherche : l'utilisateur n'est plus confronté à une unique boîte de recherche, mais à un ensemble de moteurs de recherche spécialisés. En 2007, David Hills, chef de la direction de l'entreprise LookSmart, mentionnait dans un entretien [5] une analogie entre les moteurs de recherche verticaux et la télévision câblée :

> *Like Cable was to television, vertical search is delivering to consumers relevant and essential content versus an exhaustive return on information.*

Moteurs de recherche thématiques

Les moteurs de recherche verticaux sont des moteurs de recherche caractérisés par leur spécialisation qui peut être de nature diverse, aussi bien fonctionnelle que thématique. Nous nous intéressons ici aux moteurs de recherche thématiques, c'est-à-dire traitant d'un thème ou domaine en particulier. Ces moteurs de recherche permettent de répondre à des besoins d'information précis en limitant le champ de leur index (réduction des am-

5. http://www.searchenginejournal.com/looksmart-goes-deep-vertical/2409/

biguïtés, granularité des traitements plus fine, connaissances du domaine) et en proposant des outils adaptés au domaine de recherche (interface à facettes, fonctionnalités de recherche avancées). De plus, la diminution du nombre de documents à traiter permet d'améliorer la *fraîcheur* des données ainsi que leur qualité (Web profond, indexation spécialisée).

Deux approches principales existent pour créer des moteurs de recherche thématiques : *(i)* spécialiser un moteur de recherche généraliste en modifiant la requête et/ou les résultats renvoyés par le moteur ; *(ii)* constituer un index spécialisé et monter un moteur de recherche sur ce dernier. La seconde approche permet de maîtriser la fraîcheur et la qualité des données, mais également de mieux tirer parti du domaine traité pour améliorer l'indexation et la recherche. Nous nous orientons donc naturellement vers cette approche.

Collecte de documents spécialisés

La procédure de collecte de documents spécialisés à partir du Web peut être réalisée de trois manières différentes :

1. par *recherche orientée*, c'est-à-dire l'utilisation d'un moteur de recherche pour trouver des pages pertinentes pour le thème étudié.
2. par *exploration orientée*, soit explorer le Web en orientant son parcours vers les pages pertinentes pour le thème étudié.
3. par *sélection manuelle* des pages.

La sélection manuelle de sites thématiques est un travail fastidieux et fort coûteux. Nous nous intéressons donc aux deux méthodes (semi-) automatiques existantes : la *recherche orientée* et l'*exploration orientée*.

Nos contributions

Les procédures de recherche orientée n'ont, à notre connaissance, pas été évaluées dans un cadre rigoureux et reproductible. Dans un premier temps, nous réalisons une évaluation objective et sur un grand nombre de thèmes en nous appuyant sur l'OpenDirectory, un large répertoire de sites Internet annotés manuellement. Cette évaluation nous fournit alors des performances de référence auxquelles nous comparer dans la suite de notre travail.

Fort de nos premiers résultats, nous nous intéressons à la sélection de bonnes requêtes pour la recherche orientée. Étant donné que les requêtes formulées sont des conjonctions de termes, nous proposons d'abord une

mesure de prédiction des performances (thématique) des termes sur le Web. Après avoir évalué l'impact de nombreux paramètres sur les performances de cette mesure, nous montrons que cette dernière est positivement corrélée avec la précision moyenne des termes. Nous appliquons enfin cette mesure pour sélectionner les requêtes dans un contexte de recherche orientée.

Après être intervenus en amont de la procédure de recherche orientée, nous intervenons en aval de cette dernière en ordonnant par pertinence thématique les documents téléchargés. Nous proposons de modéliser la procédure de recherche orientée sous la forme d'un graphe triparti et appliquons un algorithme de marche aléatoire pour ordonner globalement et simultanément les termes, requêtes et documents impliqués dans la procédure. Nous montrons que l'ordre fourni par notre méthode permet d'améliorer la précision de la collection produite ainsi que la sélection de termes amorces dans le cadre d'une procédure de recherche orientée itérative.

Nous nous sommes focalisés jusqu'alors sur la procédure de recherche orientée qui s'appuie sur un moteur de recherche pour déceler des documents thématiques pertinents. Une autre approche pour la collecte thématique est l'exploration orientée dont l'objectif est d'explorer le Web en orientant son parcours en direction des documents pertinents. Nous présentons notre implémentation d'un *crawler* distribué et passant à l'échelle, dénommé Babouk. Puis, nous considérons la problématique de l'ordonnancement de la frontière de *crawl* qui est la pierre angulaire de l'exploration orientée. En effet, la stratégie d'ordonnancement définit comment prioriser le téléchargement des pages et maximiser le téléchargement de pages pertinentes tout en minimisant le téléchargement de pages non pertinentes. Nous proposons d'utiliser un algorithme d'apprentissage automatique pour apprendre une fonction d'ordonnancement efficace. Cet algorithme est entraîné à partir de corpus de *crawls* existants, annotés automatiquement à l'aide de catégoriseurs thématiques. Nous comparons enfin notre *crawler* avec plusieurs *crawlers* de référence. Nous obtenons des performances proches d'une stratégie d'ordonnancement état de l'art. Ce dernier résultat nous invite à poursuivre nos recherches sur ce sujet.

Organisation du manuscrit

Ce mémoire est organisé de la manière suivante : Dans le chapitre 1, nous présentons le contexte de cette thèse : le Web (§ 1.1), la recherche d'information (§ 1.2) et les moteurs de recherche Web (§ 1.3 et 1.4). Nous nous attardons notamment sur la notion de verticalité d'un moteur de recherche avant de nous concentrer sur les moteurs de recherche thématiques. Dans

le second chapitre, nous passons en revue les méthodes existantes de collecte thématique de documents. Nous nous intéressons en particulier à deux approches majeures pour la création d'index spécialisés : la recherche orientée et l'exploration orientée.

Dans la suite du manuscrit, nous décrivons les modèles, expériences et analyses en recherche et exploration orientées menées dans le cadre de cette thèse. Nous nous focalisons tout d'abord sur la recherche orientée. Le chapitre 3 a pour objectif de décrire notre méthode d'évaluation. Les données (§ 3.1) et le protocole expérimental (§ 3.2 et § 3.3) sont présentés. Une évaluation des performances est menée à la section 3.4 et fournit des performances de référence auxquelles nous comparer lors des expériences suivantes. Nous étudions notamment les performances en fonction de deux axes : la taille des requêtes formulées (3.4.1) et les catégories étudiées (3.4.2). Les résultats obtenus montrent la nécessité d'appliquer des traitements supplémentaires pour augmenter la qualité des collections documentaires produites. Pour ce faire, nous envisageons deux approches : améliorer la formulation du besoin informationnel thématique (c'est-à-dire les requêtes thématiques) en amont du moteur de recherche, et améliorer le filtrage des documents téléchargés en aval du moteur de recherche.

Au chapitre 4, nous proposons une mesure visant à prédire les performances d'une requête. Nous commençons par définir une mesure de pertinence thématique d'un terme sur le Web (§ 4.2). Cette mesure se calcule en deux temps. Tout d'abord, un ensemble de connaissances exogènes sont collectées (4.2.1). Puis, un critère dit de cohésion thématique (4.2.2) utilise ces données pour assigner un score au terme. Nous évaluons l'impact de plusieurs paramètres sur les performances de cette mesure (4.3.1) et l'appliquons à la sélection de requêtes pour la recherche orientée (4.3.2).

Le chapitre 5 poursuit notre travail sur la recherche orientée et décrit une méthode d'ordonnancement des documents téléchargés. Notre méthode se fonde sur un modèle de graphe triparti incluant les termes amorces, les requêtes, les documents et les termes extraits de ces derniers (§ 5.2). Nous appliquons un algorithme de marche aléatoire biaisé (§ 5.3) afin d'assigner simultanément un score d'importance aux documents, aux requêtes et aux termes. Les documents peuvent ensuite être ordonnés en fonction de leur poids, ce qui permet par exemple de les filtrer plus simplement (5.4.1). L'ordre des termes peut être utilisé pour sélectionner de nouveaux termes amorces dans le cadre d'une recherche orientée itérative (5.4.2).

Le chapitre 6 est, quant à lui, dédié à l'exploration orientée du Web. Après avoir implémenté un premier *crawler* distribué et passant à l'échelle, Babouk (§ 6.1), nous proposons d'appliquer un algorithme d'apprentissage de fonctions d'ordonnancement (§ 6.2), à la croisée de la recherche d'information et de l'apprentissage statistique, pour ordonner la frontière

de crawl. Nous comparons différents algorithmes avant d'évaluer finalement celui offrant les meilleures performances en conditions réelles sur plusieurs *crawls* thématiques (6.2.2).

Contexte

Les travaux de recherche réalisés pour cette thèse s'inscrivent également dans le cadre de plusieurs projets de recherche nationaux (ANR) et européens (7ème PCRD).

PROJET METRICC

Le projet ANR *Metricc* (MÉmoire de Traduction, Recherche d'Information et Corpus Comparables) a pour objectif d'exploiter les corpus comparables (Déjean et Gaussier, 2002) dans le cadre de trois applications industrielles concrètes : les mémoires de traduction, la recherche d'information et la catégorisation interlingue. Nos travaux dans ce projet concernent plus précisément la collecte de corpus comparables spécialisés à partir du Web (chapitres 3, 4 et 5), ainsi que la catégorisation de texte interlingue.

PROJET TTC

TTC (Terminology Extraction, Translation Tools and Comparable Corpora) est un projet FP7 dont l'objectif est la génération automatique de terminologies bilingues à partir de corpus comparables dans cinq langues européennes (anglais, français, allemand, espagnol et letton) ainsi qu'en chinois et en russe. Nos travaux dans ce cadre se situent une nouvelle fois dans la création de corpus comparables spécialisés à partir du Web (chapitre 6), ainsi que dans l'exploitation de ces corpus pour l'extraction de terminologies bilingues.

PROJET SUMACC

Le projet ANR *Sumacc* propose d'explorer des stratégies d'apprentissage originales pour l'indexation de flux d'information. L'application visée est en particulier la catégorisation et la détection de catégories émergentes dans des flux multimédia (RSS, tweets, ...). Nos travaux sont ici centrés sur la catégorisation automatique de documents textuels (chapitre 6).

Première partie

MOTEURS DE RECHERCHE WEB ET COLLECTE THÉMATIQUE

1 MOTEURS DE RECHERCHE WEB

1.1 Web

> “ *WorldWideWeb: Proposal for a HyperText Project* ”
>
> Titre d'un document électronique coécrit par
> Tim Berners-Lee et Robert Caillau. 1990.

Le *World Wide Web* ou *Web* (Figure 1.1) est un système *hypertexte* inventé dans les années 1990 par Tim Berners-Lee au CERN (Organisation européenne pour la recherche nucléaire). Il est composé d'une multitude de documents identifiés *via* une URL (Uniform Resource Locator) unique et liés entre eux par des *hyperliens*. Techniquement, les pages Web sont généralement des documents semi-structurés au format HTML (HyperText Markup Language) ou XHTML (Extensible HTML). Ces derniers incluent fréquemment d'autres fichiers tels que des images, des feuilles de style, ou des scripts dynamiques.

Le *Web* est accessible en un clic *via* l'utilisation d'une connexion à Internet et d'un navigateur Web. Tout utilisateur du Web est alors en mesure de consulter, mais également de publier son propre contenu, sans nécessiter l'approbation d'une autorité particulière. Seules trois règles doivent être respectées : écrire une page au format HTML, définir une URL pour cette page et la rendre disponible sur Internet par le protocole HTTP (HyperText Transfer Protocol).

Les débuts du *Web* ont été marqués par l'arrivée de répertoires de sites créés manuellement à l'instar du répertoire Yahoo ![1]. Les créateurs de sites Web étaient alors invités à soumettre leurs URL à ces répertoires afin d'être plus facilement découverts par les utilisateurs d'Internet. Toutefois, cette approche n'a pas résisté à l'explosion du nombre de pages Web et une approche automatique s'est avérée nécessaire pour passer à l'échelle : le *crawling*. Le *crawling* consiste à naviguer de page en page, au travers des hyperliens, pour obtenir une liste des documents disponibles sur le *Web*.

1. http://dir.yahoo.com/

FIGURE 1.1 : Le premier logo du *World Wide Web* créé par Robert Cailliau en 1990.

Deux conclusions découlent de cette observation :
- Les pages qui ne sont jamais liées ne peuvent pas être découvertes [2], sauf si leur URL est connue ;
- Maintenir une liste des pages du Web à jour (ou *fraîche*) est un processus extrêmement complexe et coûteux.

1.2 Recherche d'information

> « *Information retrieval is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).* »
>
> Christopher D. Manning, Prabhakar Raghavan et Hinrich Schütze,
> An Introduction to Information Retrieval, 2008.

La *recherche d'information* (RI) est le procédé permettant à un utilisateur de trouver une réponse satisfaisant son *besoin d'information* dans une masse d'informations. Ce procédé comprend la représentation, le stockage, l'organisation et l'accès aux informations (Baeza-Yates et Ribeiro-Neto, 1999, chap. 1, p. 1).

Nous nous intéressons plus précisément à la recherche d'information assistée par ordinateur, où le procédé de recherche d'information est compris dans un système de recherche d'information, aussi nommé *moteur de recherche*. Dans ce cadre, un utilisateur formule son *besoin d'information* sous la forme d'une *requête* dans le format d'entrée du système de recherche d'information (typiquement des mots-clés et des opérateurs booléens). Cette requête est transmise au moteur de recherche qui la met en correspondance avec sa collection de *documents*, pour retourner un ensemble de résultats potentiellement pertinents pour l'utilisateur. Étant donné la masse de documents traitée par les moteurs de recherche, les systèmes de recherche d'information modernes renvoient plus généralement

2. Elles appartiennent à ce que l'on appelle le *Web invisible* (Bergman, 2001), et qui fait partie du *Web profond*.

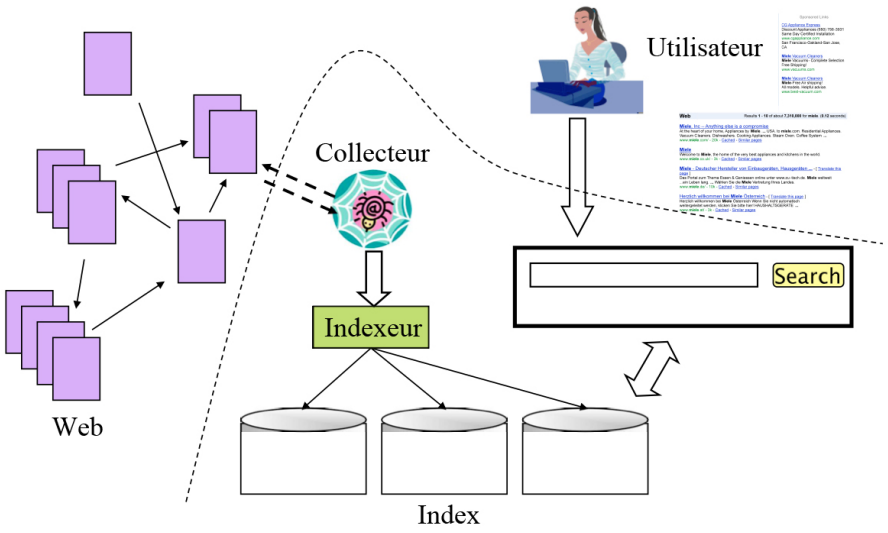


FIGURE 1.2 : Les différents composants d'un moteur de recherche Web.

une liste de documents ordonnés en fonction de leur pertinence par rapport à la requête.

Enfin, les *moteurs de recherche Web* ont la particularité de travailler sur les données issues du Web. Pour collecter ces données, ces derniers font appel à un collecteur (ou *crawler*) qui leur permet de parcourir et télécharger les pages Web. Pour permettre une recherche plus rapide parmi les documents collectés, ces derniers sont *indexés* par le moteur de recherche. Un schéma récapitulatif des différents composants d'un tel moteur est présenté à la Figure 1.2, adaptée de (Manning et coll., 2008, chap. 19, p. 434).

Les documents issus du Web possèdent également d'autres caractéristiques qui font de la recherche d'information sur le Web un véritable enjeu technologique. L'une d'elles est le format HTML utilisé pour représenter les pages Web. Ce langage de balisage permet de structurer la présentation des pages Web mais ne fournissait pas, jusqu'à récemment [3], de mécanisme clair pour spécifier la sémantique des différents constituants des documents (menu, publicité, section d'introduction, résumé...). L'une des tâches incombant au moteur de recherche est par conséquent de segmenter et quantifier l'intérêt de ces constituants pour les utilisateurs. Un autre enjeu pour les moteurs de recherche Web est la détection des pages et sites de *spam* : avec l'augmentation du nombre d'utilisateurs du Web, une plus grande exposition sur le Web est devenue synonyme d'une augmentation des retombées économiques. Les moteurs de recherche, consultés

3. La version 5 du langage HTML, encore en cours d'élaboration, fournira plusieurs balises sémantiques permettant de structurer le contenu des documents Web (http://www.w3.org/TR/html5/sections.html).

quasi systématiquement lors de la plupart des navigations Web, sont naturellement devenus des cibles privilégiées des « spammeurs » qui tentent délibérément d'améliorer la position de leurs sites de manière injustifiée pour capter toujours plus d'utilisateurs (Gyöngyi et Garcia-Molina, 2005).

1.3 Moteurs de recherche verticaux

Un moteur de recherche vertical, par opposition à un moteur de recherche généraliste ou horizontal, est un moteur de recherche caractérisé par sa spécialisation, son périmètre. La définition exacte de la *verticalité* ou spécialisation d'un moteur de recherche vertical reste pourtant floue, aussi bien dans le domaine industriel qu'académique. À titre d'exemple, les articles scientifiques traitant de *recherche d'information fédérée* (ou méta-recherche) ou de *recherche d'information agrégée* (Arguello et coll., 2009; Diaz et coll., 2010; Murdock et Lalmas, 2008; Arguello et coll., 2011; Shokouhi et Si, 2011) mentionnent de nombreux moteurs de recherche verticaux de natures diverses, dont voici quelques exemples : actualités (*news*), articles scientifiques (*scholar*), *blogs*, cartes (*maps*), connaissances encyclopédiques, images, forums de discussion, livres, local, média sociaux, offres d'emplois, produits (e-commerce), profils d'artistes ou d'entreprises, et vidéos.

Nous proposons une première classification des verticalités rencontrées en six grandes catégories :

1. Verticalité en format :
   - Format de fichier (HTML, PDF, DOC)
2. Verticalité en genre :
   - Type de média (cartes, images, vidéos)
   - Type de site (blog, forum, livre, réseaux sociaux, *wikis*)
   - Type de texte (brevets, articles scientifiques, journalistiques)
3. Verticalité géographique :
   - Pays de publication
   - Localité de l'utilisateur
4. Verticalité langagière :
   - Langue du document
   - Langue simplifiée
5. Verticalité temporelle :
   - Date/Heure de publication
   - Recherche en temps réel
6. Verticalité thématique :
   - Thème (cuisine, emploi, immobilier, juridique, médical, musique, voyage)

La verticalité d'un moteur de recherche vertical correspond donc, de manière générale, à un périmètre limité, quel qu'il soit. Nous allons nous intéresser plus particulièrement aux moteurs de recherche verticaux thématiques (ou simplement moteurs de recherche thématiques), c'est-à-dire des moteurs de recherche traitant un thème (ou domaine) en particulier. Ces moteurs sont particulièrement intéressants, car ils fournissent des points d'entrée thématiques au Web. De plus, couplés à d'autres informations du même domaine (actualités, réseaux sociaux), ces *portails* offrent aux utilisateurs un site Web de référence auprès duquel s'informer quotidiennement sur les sujets qui les intéressent, à la manière des chaînes de télévision thématiques [4].

1.4 Moteurs de recherche thématiques

Les moteurs de recherche thématiques, moteurs de recherche spécialisés (dans un domaine ou un thème), parfois nommés portails de recherche (verticaux) ou « vortails », existent depuis les débuts du Web. À titre d'exemple, le moteur de recherche *Google* proposait dès 1999 des moteurs de recherche thématiques sur des thèmes aussi divers que les systèmes d'exploitation (google.com/linux, google.com/mac, ...) ou les informations gouvernementales américaines (google.com/unclesam). En complément de ces exemples, plusieurs moteurs de recherche thématiques en langues française et anglaise sont présentés au Tableau 1.1.

De même, dans le milieu académique, les campagnes d'évaluation témoignent d'un passé de recherche de plus de dix ans en recherche d'information spécialisée. La campagne d'évaluation CLEF (*Conference and Labs of the Evaluation Forum* [5]) a ainsi proposé une tâche dédiée à la recherche d'information spécialisée (*domain-specific IR*) entre 2000 et 2009 (Kluck et Gey, 2001; Kluck, 2004). La campagne d'évaluation TREC (Text REtrieval Conference), bien que n'ayant pas proposé de tâche de recherche d'information spécialisée, a encapsulé un certain nombre de tâches sur des domaines spécialisés tels que la chimie (*TREC Chemistry IR*, 2009-2011), la bio-informatique (*TREC Genomics*, 2003-2007), la santé (*TREC Medical Records*, 2011-2012) ou la justice (*TREC Legal*, 2006-2012).

4. http://www.searchenginejournal.com/looksmart-goes-deep-vertical/2409/
5. ou *Cross-Language Evaluation Forum* jusqu'en 2009.

Thème	Nom	URL
Chimie	ChemBioFinder	http://www.chemfinder.com/
	ProfusionChimie	http://www.profusion-chimie.1s.fr/
Cuisine	RecipeBridge	http://www.recipebridge.com/
	Yummly	http://www.yummly.com/
Génétique	@Genetics	http://atgenetics.com/
	Genentech	http://www.gene.com/
Immobilier	eListIt	http://www.elistit.com/
	Estately	http://www.estately.com/
	Trulia	http://www.trulia.com/
Ingénierie	BuildingOnline	http://www.buildingonline.com/
	GlobalSpec	http://www.globalspec.com/
Juridique	EasyDroit	http://www.easydroit.fr/
	e-Justice	http://www.ejustice.fr/
	LexisNexis	http://www.lexisnexis.com/
Médical	HealthLine	http://www.healthline.com/
	MedlinePlus	http://www.nlm.nih.gov/medlineplus/
	MedStory	http://www.medstory.com/
	PubMed	http://www.ncbi.nlm.nih.gov/pubmed/
	Quertle	http://www.quertle.info/
Physique	PhysLink	http://www.physlink.com/
Scientifique	CiteSeer	http://citeseerx.ist.psu.edu/
	Isidore	http://rechercheisidore.fr/
	SciRus	http://www.scirus.com/
	Science.gov	http://www.science.gov/
	sciseek	http://www.sciseek.com/

Tableau 1.1 : Exemples de moteurs de recherche thématiques industriels

Étant donné leur périmètre limité, nous pouvons espérer retirer plusieurs avantages des moteurs de recherche thématiques :

1. Tout d'abord, le thème du moteur réduit les ambiguïtés de types homonymiques ou sémantiques. Le terme *virus*, par exemple, peut référer à un virus informatique ou biologique. Face à ce type de requêtes, les moteurs de recherche généralistes n'ont d'autre alternative que de présenter toutes les propositions aux utilisateurs, soit implicitement en maximisant la *diversité* des résultats (Clarke et coll., 2009), soit explicitement (Carpineto et coll., 2009). Les moteurs de

recherche thématiques, au contraire, sont porteurs d'un contexte qui permet d'améliorer l'interprétation des requêtes utilisateur.

2. La réduction du nombre de documents à traiter permet d'améliorer la *fraîcheur* des données collectées.
3. La spécialisation du moteur permet une intégration de données issues du Web profond, de données structurées et de connaissances du domaine (ontologies) à différents niveaux du moteur de recherche.
4. L'interface homme-machine peut tirer parti du thème pour proposer des outils de recherche uniques, plus performants que les outils génériques d'un moteur de recherche généraliste. Sur un site de recettes de cuisine, il est par exemple intéressant de se voir proposer des outils pour sélectionner certains temps de cuisson, types de plats ou familles d'ingrédients.
5. Les utilisateurs sont souvent des spécialistes du domaine qui feront usage de fonctionnalités de recherche avancées.

Steele (2001) a proposé une première taxinomie des moteurs de recherche spécialisés. Nous en proposons une version revue et améliorée à la Figure 1.3 [6]. Les deux grandes approches pour la création de moteurs de recherche spécialisés sont de spécialiser un moteur de recherche généraliste en modifiant la requête et/ou les résultats renvoyés par le moteur, ou de créer un moteur de recherche sur un index spécialisé.

La très grande majorité des travaux visant à spécialiser un moteur de recherche généraliste s'appuie sur l'ajout de mots-clés contextuels afin de spécialiser les requêtes utilisateurs. Les requêtes modifiées sont ensuite soumises à un moteur de recherche généraliste et les résultats de ce dernier sont présentés à l'utilisateur, après un éventuel filtrage automatique. Oyama et coll. (2001) extraient automatiquement ces mots-clés, qu'ils nomment *keyword spices*, à partir de documents du domaine d'intérêt et d'un arbre de décision. Ces mots-clés sont ensuite ajoutés automatiquement aux requêtes utilisateurs et soumis à un moteur de recherche. Glover et coll. (2001) utilisent une approche similaire, mais s'appuient sur un catégoriseur fondé sur un séparateur à vaste marge (SVM) pour sélectionner des termes saillants thématiques. Sondhi et coll. (2010) choisissent leurs mots-clés contextuels à partir de traces de requêtes utilisateurs (*query logs*) issues d'un moteur de recherche généraliste. Tang et coll. (2006, 2009)

6. En ce qui concerne la construction d'un index spécialisé, la *sélection manuelle* de sites à indexer a été ajoutée. Au contraire, nous avons supprimé les *méthodes de crawling au moment de la requête*, peu usitées. Nous avons renommé *métarecherche* en *recherche orientée*. Pour les méthodes utilisant un moteur de recherche, nous avons séparé les *méthodes de spécialisation de requête* et de *réordonnancement*.

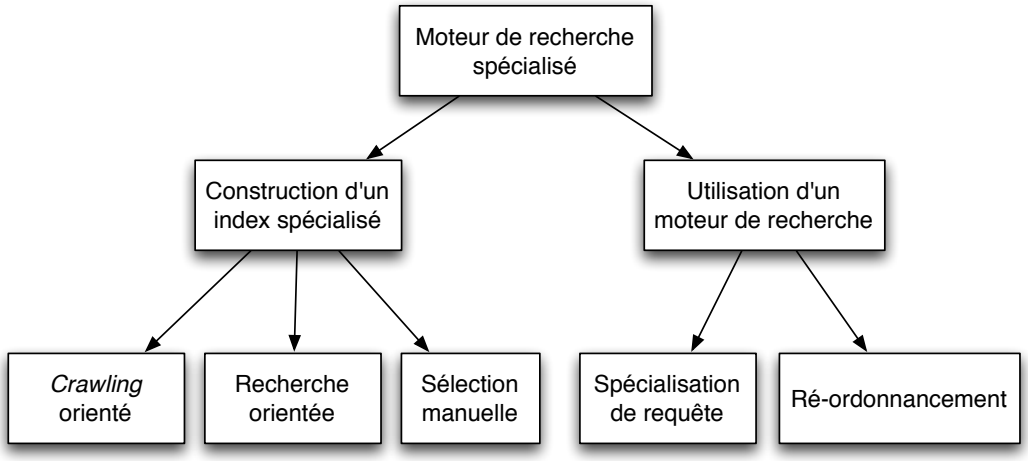


FIGURE 1.3 : Taxinomie des moteurs de recherche spécialisés.

étudient le domaine de la santé mentale et obtiennent des performances surprenantes en ajoutant simplement l'intitulé du thème (« *depression* ») à toutes les requêtes. Le moteur de recherche PubMed propose des « stratégies de recherche » créées semi-automatiquement qui ajoutent un ensemble de mots-clés aux requêtes utilisateur[7]. Enfin, le projet Watson (Budzik et Hammond, 1999) entre également dans ce courant et offre un outil de recherche qui modifie les requêtes d'un utilisateur pour y intégrer un contexte de recherche pouvant être thématique.

La seconde grande approche pour la création de moteurs de recherche spécialisés est de créer ou d'utiliser un index spécialisé. Cette approche permet de maîtriser la fraîcheur et la qualité des données, mais également de mieux tirer parti du domaine traité pour améliorer l'indexation et la recherche. La plupart du temps, de larges collections documentaires spécialisées existent (collection d'articles médicaux, de brevets, ...). Néanmoins, dans le cas des moteurs de recherche Web, il est nécessaire de trouver et de sélectionner les documents à partir du Web. Nous allons, dans le chapitre 2, nous concentrer sur la problématique de sélection de pages Web pour la construction d'index spécialisés.

7. http://www.nlm.nih.gov/bsd/special_queries.html

2 COLLECTE THÉMATIQUE DE DOCUMENTS

2.1 Présentation

Pour construire un index spécialisé à partir du Web, il est nécessaire de collecter un ensemble de documents thématiques. Comme nous l'avons vu à la Figure 1.3, nous pouvons distinguer trois approches :

1. Le *crawling orienté* (*Focused crawling*), soit explorer le Web en orientant son parcours vers les pages pertinentes pour le thème étudié.
2. La *recherche orientée* (*Focused search*), c'est-à-dire l'utilisation d'un moteur de recherche pour trouver des pages pertinentes pour le thème étudié.
3. La *sélection manuelle* des pages.

La sélection manuelle de sites thématiques est une méthode fréquemment utilisée, mais qui nécessite un travail fastidieux et a un coût élevé. Nous allons plutôt nous intéresser aux deux méthodes (semi-)automatiques existantes. Dans la section suivante, nous étudierons l'état de l'art en recherche orientée. Puis, à la section 2.3, nous examinerons les travaux existants sur l'exploration orienté. Enfin, au vu des travaux réalisés, nous terminerons ce chapitre par une synthèse récapitulant les points majeurs de chacune des approches.

2.2 Recherche orientée

Les moteurs de recherche Web fournissent un point d'entrée commode à la masse de connaissances disponible sur le Web. En effet, ces moteurs de recherche permettent, à un coût minimal, d'accéder à un index du Web avec une couverture très large et dont une grande partie des pages de *spam* a été filtrée.

La Figure 2.1 synthétise le fonctionnement de cette approche. Le thème est représenté sous la forme d'une ou plusieurs requêtes suivant le format d'entrée du moteur de recherche. Les requêtes sont ensuite soumises à ce dernier et les pages Web résultantes sont enfin téléchargées et stockées.

La communauté de recherche d'information s'est jusqu'à présent peu intéressée à cette approche. Notons que, parmi les travaux cités précédemment, certaines méthodes de création de requêtes thématiques sont

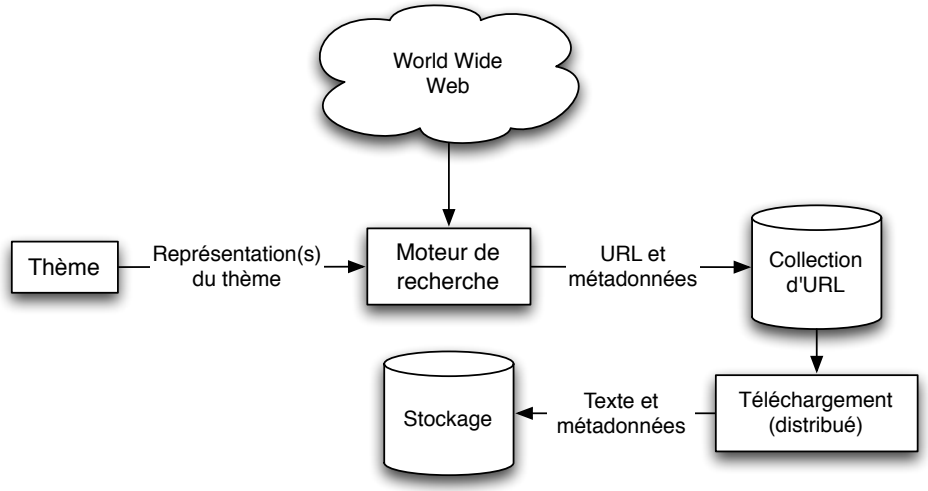


FIGURE 2.1 : Représentation haut niveau de l'architecture d'un système de collecte basé sur un moteur de recherche Web.

fonction du thème et non des requêtes utilisateurs et peuvent donc être appliquées pour constituer des index spécialisés (Glover et coll., 2001; Oyama et coll., 2001).

En revanche, le domaine du traitement automatique des langues s'est particulièrement intéressé à l'utilisation des moteurs de recherche Web comme source de *corpus* (collections de documents). C'est le *Web as a Corpus* (Kilgarriff et Grefenstette, 2003).

Brin (1999); Riloff et Jones (1999); Agichtein et Gravano (2000), entre autres, se sont par exemple intéressés à l'extension d'un petit ensemble de relations entre entités *via* le Web. Ces systèmes faiblement supervisés, dits de *bootstrapping*, démarrent à partir de quelques exemples. Puis, itérativement, ils formulent des requêtes pour trouver de nouveaux contextes d'apparition des entités sur le Web, et utilisent les contextes appris pour extraire de nouvelles entités et relations.

Fletcher (2001) et Renouf et coll. (2006) se sont intéressés à créer des concordanciers [1] du Web à partir des moteurs de recherche. Pour ce faire, les requêtes des utilisateurs sont simplifiées et transmises aux moteurs de recherche Web. Puis, les résultats de ces derniers sont téléchargés et analysés à la volée dans le but d'offrir de nouveaux opérateurs et moyens de filtrage aux utilisateurs linguistes.

1. Un concordancier est un logiciel qui affiche les concordances d'un mot, c'est-à-dire les occurrences d'un mot avec leurs contextes d'apparition dans un texte ou un ensemble de textes.

Plus proches de nos problématiques, Ghani et coll. (2000; 2005) se sont intéressés à la création automatique de corpus et de modèles de langue pour les langues minoritaires (comme, par exemple, le Tagalog) à partir du Web. Leur système démarre à partir d'un corpus en langue minoritaire et est composé de deux modules : un module de génération de requêtes et un module de détection de langue. Dans un premier temps, une requête est créée à partir d'une combinaison de mots-clés extraits du corpus fourni en entrée. Cette requête est soumise à un moteur de recherche et les pages Web sont téléchargées et nettoyées de leurs balises. Puis, le détecteur de langue est appliqué aux documents obtenus afin de filtrer ceux n'appartenant pas à la langue étudiée. Enfin, l'outil de génération de requête « apprend » la qualité de la requête générée précédemment en comptant la proportion de documents téléchargés appartenant effectivement à la langue voulue. Le système peut ainsi générer de nouvelles requêtes faisant intervenir les mots les plus pertinents.

Enfin, Baroni et Bernardini (2004) ont proposé BootCaT, une procédure de création semi-automatique de corpus et lexiques spécialisés (Figure 2.2). La procédure BootCaT est démarrée par un petit nombre de termes amorces (généralement entre 5 et 15 termes). Ces termes sont combinés aléatoirement en requêtes, qui sont soumises à un moteur de recherche. Les documents renvoyés par le moteur de recherche sont ensuite téléchargés et analysés. Contrairement à Ghani et coll. (2005), seul le « contenu informatif » est extrait, c'est-à-dire le contenu textuel de la page, privé de tous les éléments incohérents avec le contenu de la page. Ces éléments incluent les informations de navigation, listes de liens internes et externes, publicités, copyright et mentions légales, hauts et bas de page, éléments répétés sur tout/une partie du site, et les éléments de *spam* tels que des commentaires postés automatiquement sur des blogs[2]. Une validation manuelle est appliquée après chaque itération, permettant d'assurer la bonne qualité des termes et documents sélectionnés.

La procédure BootCaT ayant pour but la création de corpus textuels, une série de filtres supplémentaires est appliquée afin d'obtenir des textes de bonne qualité. Ainsi, les documents dont la taille n'excède pas les 5 Ko ou dépasse les 200 Ko sont ignorés, ce qui permet de filtrer facilement certains types de pages de *spam* ou d'erreurs (Fletcher, 2004). De plus, les documents doublons (*near-duplicates*), nombreux sur le Web, sont détectés et supprimés grâce à une version simplifiée de l'algorithme de Broder (Broder, 2000; Baroni et Ueyama, 2006).

2. Cette définition est une traduction de celle fournie aux annotateurs dans le cadre de la campagne d'évaluation CLEANEVAL :
http://cleaneval.sigwac.org.uk/annotation_guidelines.html

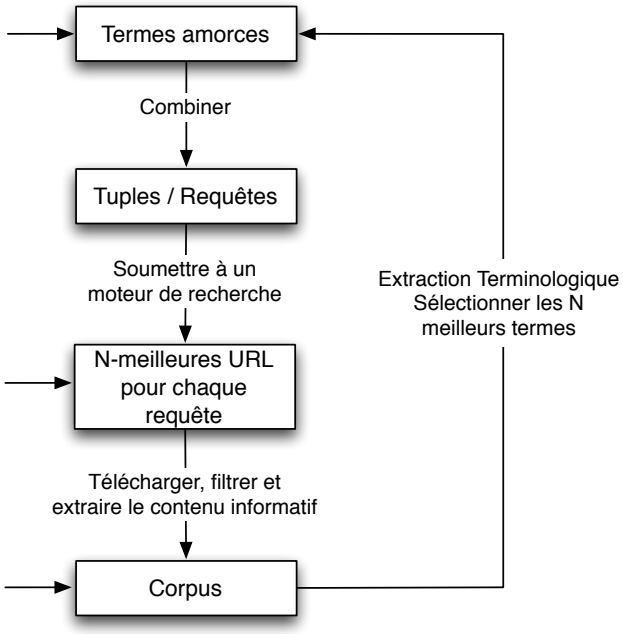


FIGURE 2.2 : Procédure BootCat pour la constitution de corpus et de terminologies à partir du Web.

En 2005, le groupe ACL SIGWAC (The Special Interest Group of the ACL on Web as Corpus) est créé. Il fédère dès lors les travaux sur le *Web as Corpus* notamment au travers de leur atelier annuel WAC. La procédure BootCaT est appliquée pour créer de larges corpus spécialisés et généraux (Sharoff, 2006) dans une grande variété de langues (Baroni et coll., 2009).

2.3 Exploration orientée

Comme nous l'avons vu au chapitre précédent, utiliser un moteur de recherche comme point d'entrée au Web offre plusieurs avantages :

- Simple : ne nécessite qu'un ordinateur personnel et un accès à Internet
- Efficace : collecte rapide
- Prétraité : documents indésirables (*spam*) préfiltrés
- Couverture : index du Web

En contrepartie, l'utilisation d'un moteur de recherche Web impose les limites suivantes :

- Éléments inconnus : critères de filtrage, critères de tri des résultats
- Limites : nombre de résultats renvoyés, nombre de requêtes soumises par jour, langues disponibles

– Maintien à long terme : fermeture possible des API d'accès aux moteurs de recherche [3]
– Expressivité : spécification du besoin basée uniquement sur des requêtes de type « mots-clés »

Le *crawling* orienté a un coût plus important et nécessite de répliquer les traitements réalisés par les moteurs de recherche Web (*spam*, qualité des pages, *crawling* distribué). Néanmoins, il permet un plus grand contrôle du processus de collecte.

2.3.1 *Exploration*

Les *crawlers* font partie intégrante des moteurs de recherche Web et sont dédiés à la collecte et à l'indexation des documents Web. Ils sont également utilisés dans un but d'archivage [7], d'extraction d'information (pour trouver des adresses e-mail ou des pages de produits par exemple) et de constitution de corpus à la demande (Baroni et Ueyama, 2006), entre autres.

Un processus de *crawl* (Figure 2.3, adaptée de (Castillo, 2005)) démarre à partir d'URL amorces (*seed URL*). Les pages Web correspondant à ces URL sont téléchargées et les hyperliens présents sur ces dernières sont extraits et ajoutés à un ensemble d'URL à visiter, également appelé la *frontière de crawl*. Le nombre d'URL peuplant la frontière de *crawl* augmentant très rapidement, un critère permettant de prioriser le téléchargement de certaines pages est généralement appliqué. Tour à tour, les URL les mieux classées parmi la frontière de *crawl* sont téléchargées et de nouveaux liens en sont extraits. Ce processus, en apparence simple, amène néanmoins un certain nombre de défis techniques : comment ordonner la frontière de *crawl*, comment maximiser l'utilisation de la bande passante, comment esquiver les pièges tendus aux *crawlers* (pages Web de *spam*, boucles de liens infinis), ou encore comment éviter de surcharger les serveurs Web.

Castillo (2005) a défini le comportement d'un *crawler* Web comme une combinaison de quatre principes :
– Un *principe de politesse* qui définit comment éviter de surcharger les serveurs Web.

3. Ce point est loin d'être anecdotique : il est survenu deux fois durant cette thèse (fermeture de Yahoo ! BOSS [4] puis de Bing Search API [5]. Google API [6] avait par ailleurs déjà fermé précédemment).
4. http://www.ysearchblog.com/2010/10/08/bossv2/
5. http://www.bing.com/community/site_blogs/b/developer/archive/2012/04/12/bing-dev-update.aspx
6. http://googlecode.blogspot.fr/2010/11/introducing-google-apis-console-and-our.html
7. http://www.archive.org

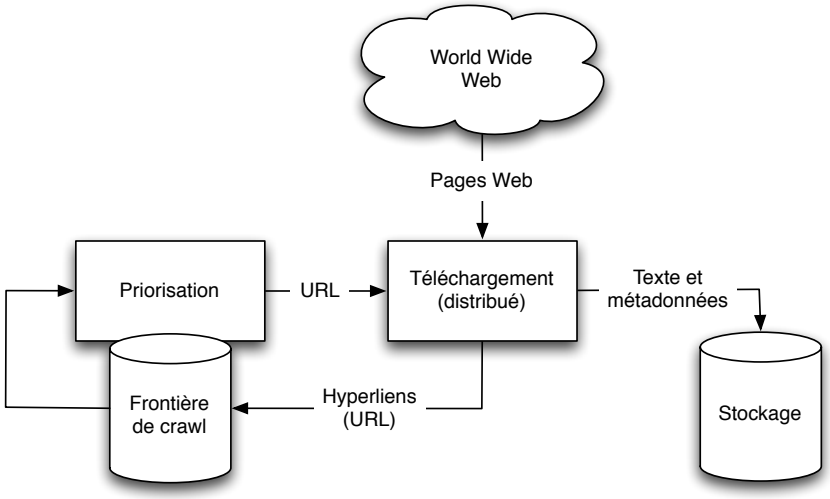


FIGURE 2.3 : Représentation haut niveau de l'architecture d'un *crawler* Web.

– Un *principe de parallélisation* qui définit comment coordonner les robots d'indexation distribués.
– Un *principe de sélection* qui définit quelles pages télécharger.
– Un *principe de revisite* qui définit quand vérifier si les pages ont été mises à jour.

Le principe de politesse est un composant obligatoire d'un *crawler* qui spécifie de quelle manière il doit se comporter face à un serveur Web. Des règles explicites (*robots.txt*) et des règles implicites régulent la fréquence à laquelle il est possible de télécharger les documents issus d'un même serveur. Le principe de parallélisation a pour objectif de maximiser la vitesse de téléchargement tout en minimisant le surcoût dû à la parallélisation du téléchargement et en évitant de télécharger la même page plusieurs fois. Cho et Garcia-Molina (2002) mentionnent deux approches : l'approche dynamique qui consiste à allouer de nouvelles URL aux *crawlers* dynamiquement, et l'approche statique qui consiste à définir comment répartir les URL avant le *crawl*. Le principe de revisite est discuté en section 2.5.

Nous allons nous intéresser plus particulièrement aux *crawlers orientés* visant à télécharger un nombre restreint de pages répondant à un besoin en un minimum de temps. Dans ce cadre, le problème d'ordonnancement de la frontière de *crawl* est certainement le plus important (principe de sélection). Contrairement aux *crawlers* généralistes, qui utilisent des mesures de l'importance des pages telles que le degré entrant (Cho et coll., 1998), le PageRank (Cho et coll., 1998), la taille du site Internet (Baeza-Yates et coll., 2005) ou l'impact sur le moteur de recherche (Pandey et Olston, 2008), les

crawlers orientés font au contraire usage de critères spécifiques au besoin, pour sélectionner les URL à télécharger en priorité.

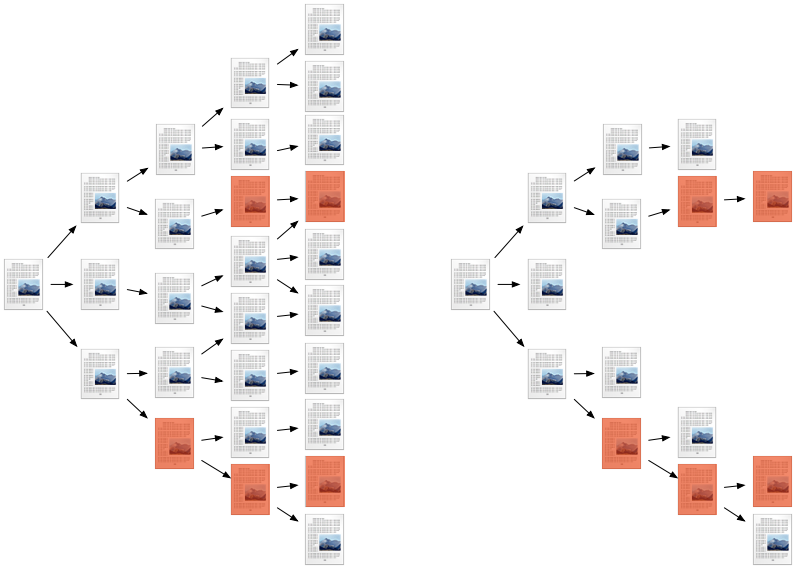
2.3.2 *Exploration orientée*

GÉNÉRALITÉS

Un *crawler* orienté (Chakrabarti et coll., 1999), aussi nommé *crawler* thématique (Pant et Menczer, 2002), est un logiciel qui explore automatiquement le Web et dont l'objectif est le rapatriement efficace de documents pertinents pour un domaine défini. Contrairement aux robots d'indexation des moteurs de recherche généralistes, ces *crawlers* nécessitent bien moins de ressources pour le parcours et le stockage des données collectées (Chakrabarti et coll., 1999). De plus, ces derniers permettent un rafraîchissement de l'index du moteur de recherche beaucoup plus fréquent (Diligenti et coll., 2000). Le fonctionnement d'un *crawler* traditionnel consiste à effectuer un parcours en largeur sur les liens sortants d'un ensemble de pages initiales et à répéter le processus sur les nouvelles pages rapatriées. Au contraire, un *crawler* orienté a pour objectif de minimiser le nombre de pages visitées qui ne sont pas en rapport avec le domaine étudié (Figure 2.4). Kumar et coll. (2000) estimaient en 2000 le nombre moyen de liens sortant pour une page internet à 7,2. Afin d'obtenir de bonnes performances, un *crawler* orienté devra donc trier les liens sortants à la frontière de *crawl* et tenter de limiter le téléchargement de pages à des pages pertinentes pour l'utilisateur. C'est la *stratégie d'ordonnancement* du *crawler* (*crawl frontier ordering* (Cho et coll., 1998)).

Les stratégies d'ordonnancement de la frontière de *crawl* peuvent être divisées en trois générations (Olston et Najork, 2010). La stratégie initiale, dénommée *fish search* (De Bra et Post, 1994), consiste simplement à évaluer la pertinence de chaque page téléchargée et à explorer les descendants de ces pages jusqu'à une certaine profondeur en fonction de la pertinence des pages rencontrées. En d'autres termes, cette approche propage la pertinence d'une page parente sur ses descendants de manière uniforme.

Au contraire, les stratégies d'ordonnancement de seconde génération (Cho et coll., 1998; Chakrabarti et coll., 1999; Abiteboul et coll., 2003; Hersovici et coll., 1998; Guan et coll., 2008; Diligenti et coll., 2000; Pant et Srinivasan, 2006) explorent les descendants des pages pertinentes de manière non uniforme, optant pour les liens les plus pertinents d'abord. Ces travaux utilisent aussi bien des indices issus du graphe de liens constitué jusqu'alors (degré entrant, PageRank, Double Focused PageRank, HITS/B-HITS, WPSS, OPIC) que le contenu des pages (texte des ancres, texte au-



(a) Crawl en largeur (b) Crawl orienté

FIGURE 2.4 : Illustration de la différence entre un *crawl* en largeur opéré par les moteurs de recherche généralistes et un *crawl* orienté.

tour des liens, texte de la page parente).

Enfin, la troisième génération de stratégies se fonde sur des méthodes d'apprentissage automatique et d'analyse de liens (Chakrabarti et coll., 2002; Diligenti et coll., 2000; Rennie et McCallum, 1999). Menezes (2004) propose d'entraîner son classifieur par apprentissage actif, algorithme itératif mettant l'utilisateur à contribution pour sélectionner bons et mauvais documents. Chakrabarti et coll. (2002) et Rennie et McCallum (1999) utilisent un apprentissage par renforcement afin d'entraîner itérativement un classifieur à reconnaître bons et mauvais liens sortants : le classifieur sera pénalisé lorsqu'il acceptera une page et que celle-ci s'avérera, une fois téléchargée et analysée, non pertinente. Au contraire, il sera récompensé lorsqu'il sélectionnera des pages pertinentes.

En parallèle de cette structuration en trois générations des stratégies d'ordonnancement, plusieurs travaux se sont intéressés à la modélisation du processus de *crawling* orienté comme système multi agents. Baujard et coll. (1998) proposent une approche multi agents où plusieurs agents spécialisés dans différents sous-domaines d'internet collaborent pour filtrer les pages non pertinentes *via* un indice de similarité entre documents amorces et documents téléchargés. De même, InfoSpiders (Pant et Menczer, 2002) (précédemment nommé ARACHNID (Menczer, 1997)) fait usage d'un ensemble d'agents constitués d'un réseau de neurones et de mots-clés.

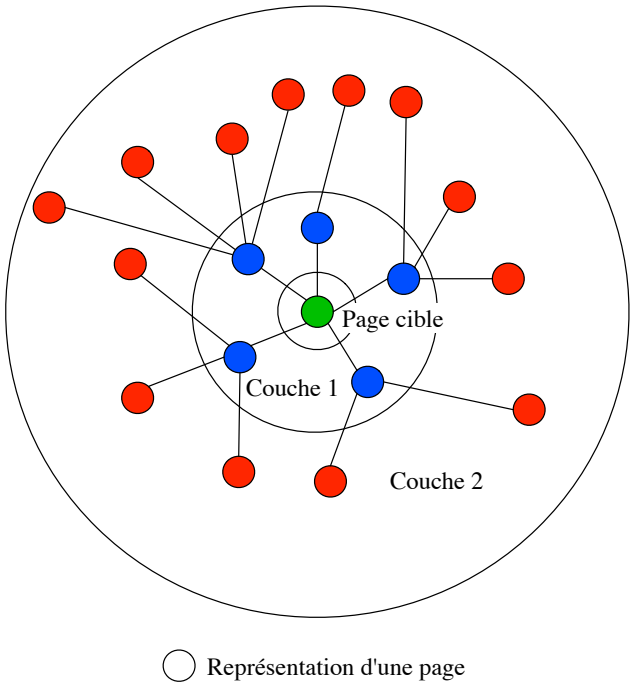


FIGURE 2.5 : Graphe contextuel d'une page cible défini par Diligenti et coll. (2000)

Pour chaque lien rencontré, les fréquences des mots-clés autour des liens sont fournies au réseau de neurones, qui détermine s'il faut ou non suivre le lien.

EFFET TUNNEL

L'intuition selon laquelle les liens sortants d'une page non pertinente ne devraient pas être analysés est remise en cause par l'effet tunnel (ou *tunneling*) qui peut être formulé comme suit : partant d'une page pertinente pour l'utilisateur, l'accès à d'autres pages pertinentes peut passer par des pages moins pertinentes. Prenons pour exemple le thème des *modèles graphiques probabilistes* : pour trouver des pages pertinentes sur ce domaine, il peut s'avérer nécessaire de traverser des pages d'universités, de groupes de recherche ou de chercheurs pour collecter enfin des pages de cours ou des articles scientifiques présentés sur les pages des chercheurs de cette communauté. La prise en compte de l'effet tunnel aura donc pour effet d'augmenter le rappel du *crawler*, ceci éventuellement au détriment de sa précision.

Pour prendre en compte ce phénomène, l'algorithme original *fish search* propage le poids des pages parentes aux pages filles avec un facteur d'amortissement. Chen et coll. (2010) utilisent une profondeur constante. Diligenti et coll. (2000) modélisent le contexte d'apparition des pages à trouver. Ils

créent un *graphe contextuel* (Figure 2.5) qui leur permet ensuite de prédire la distance séparant la page courante d'une page pertinente. La création de ce graphe est réalisée par une méthode originale s'appuyant sur un moteur de recherche traditionnel : pour une URL, les liens entrants (fournis par le moteur de recherche) jusqu'à une distance N de la page courante sont récupérés. Un ensemble de classifieurs bayésiens naïfs est ensuite entraîné sur les différentes couches (0 à N) de pages rapatriées. Finalement, ces classifieurs sont utilisés pour associer les nouvelles pages à une couche et ainsi prédire l'apparition de nouvelles pages pertinentes à N liens de la page courante. Babaria et coll. (2007) a proposé de remplacer l'ensemble de classifieurs par une unique méthode de régression ordinale. En parallèle du *crawling* orienté, Qin et coll. (2004) soumettent des requêtes à plusieurs moteurs de recherche pour atteindre les pages pertinentes distantes de la région de *crawl*. Leurs requêtes sont composées de termes spécialisés choisis manuellement par des experts du domaine. Enfin, Menczer et coll. (2003) proposent une métaheuristique simple pour équilibrer la gloutonnerie des *crawlers* : plutôt que d'ordonner systématiquement la frontière de *crawl* et de télécharger les pages les mieux pondérées, les auteurs proposent un fonctionnement par lot en téléchargeant N pages avant de réordonner la frontière de *crawl* et de générer N nouvelles pages à télécharger. Cette approche force ainsi le *crawler* à télécharger des pages moins pertinentes augmentant l'exploration du *crawler*.

Bergmark et coll. (2002) ont réalisé une étude statistique de l'effet tunnel sur un *crawl* orienté de 500 000 pages. Les auteurs ont évalué la pertinence de chaque page en calculant la similarité cosinus entre cette page et le centroïde d'un ensemble de pages de référence. Puis, ils ont calculé une moyenne de ces similarités pour toutes les pages se trouvant à une même distance d'une page pertinente. Leurs résultats, présentés au Tableau 2.1, montrent que la similarité moyenne avec le centroïde des pages de référence reste relativement stable quelle que soit la distance à la page pertinente la plus proche. De plus, la 4^{e} colonne du tableau montre que des pages à forte similarité, donc potentiellement pertinentes, apparaissent à tous les niveaux. Ces deux résultats montrent que la prise en compte de l'effet tunnel est pertinente, mais peut nécessiter de suivre un long chemin de pages non pertinentes pour aboutir à une page pertinente. Leur étude conclut également sur la nécessité d'utiliser l'historique de *crawl* ainsi que le chemin complet ayant permis d'arriver sur une page afin de décider ou non de la traverser.

ÉVALUATION

Chakrabarti, Srinivasan, Pant et Menczer se sont également intéressés à l'évaluation des stratégies d'ordonnancement. Chakrabarti et coll. (2002)

Distance	Nombre de documents	Similarité moyenne	Similarité maximale
1	16 422	0,1520	0,877
2	18 106	0,2008	0,894
3	12 699	0,1740	0,828
4	26 356	0,1536	0,798
5	49 018	0,1460	0,857
6	62 728	0,1559	0,826
7	82 287	0,1547	0,853
8	109 297	0,1383	0,855
9	59 370	0,1427	0,859
10	36 672	0,1336	0,828
11	12 390	0,1479	0,801
12	5 981	0,0926	0,627
13	6 604	0,1536	0,627
14	1 485	0,1913	0,706

Tableau 2.1 : Évolution de la similarité moyenne et maximale en fonction de la distance à une page pertinente.

estime la qualité d'un *crawler* orienté en fonction d'un critère nommé taux de récolte (*Harvest rate*) qui correspond au nombre de pages pertinentes téléchargées par rapport au nombre total de pages téléchargées. Ils s'appuient sur une taxinomie existante (OpenDirectory [8], Yahoo ! Directory [9]) et construisent un ensemble de classifieurs (un par nœud de la taxinomie) à partir de bons et mauvais exemples. Ils utilisent ensuite ces mêmes classifieurs pour évaluer la pertinence des pages téléchargées au cours du temps. Pant et Srinivasan (2006) évaluent leurs stratégies de *crawl* en se basant sur le corpus de l'OpenDirectory. Ils utilisent 20 URL d'une catégorie du répertoire comme pages amorces et les URL restantes comme pages pertinentes de référence. Srinivasan et coll. (2005) utilisent également l'OpenDirectory mais appliquent une méthodologie légèrement plus complexe. Considérant un ensemble d'URL associé à une catégorie de l'OpenDirectory (dénommées URL cibles), ils remontent à une distance de D liens de ces URL pertinentes *via* un moteur de recherche. Puis, partant de ces nouvelles URL, ils évaluent la capacité de différentes stratégies à aboutir aux URL cibles en fonction du temps. Cette tâche contrainte permet de garantir l'apparition d'au moins une page pertinente à une distance D des URL de départ et permet une évaluation relativement rapide des stratégies de *crawl*. Leur évaluation s'appuie sur trois critères :

8. http://www.dmoz.org/
9. http://dir.yahoo.com/

– Précision à N pages téléchargées (également nommée taux de récolte ou *Harvest Rate*)
– Rappel à N pages téléchargées (Pant et Menczer, 2003)
– Similarité lexicale par rapport à une « description du thème » calculée automatiquement à partir de l'OpenDirectory

2.4 Revisite

Afin de maintenir leur index à jour, les moteurs de recherche doivent surveiller continuellement si les documents qu'ils ont collectés ont été mis à jour. C'est ce que l'on nomme le re-*crawling* ou la revisite. Plusieurs travaux se sont intéressés à cette problématique dans le cadre de moteurs de recherche généralistes et peuvent être directement appliqués aux moteurs de recherche spécialisés, c'est pourquoi nous ne ferons qu'un rapide survol des travaux existant à ce sujet.

L'un des objectifs du principe de re-*crawling* est de maximiser la fraîcheur de l'index en estimant la fréquence de changement des pages de l'index du moteur de recherche pour planifier une nouvelle visite de ces dernières au moment opportun. Le re-*crawling* dépend de deux facteurs : un estimateur de la fréquence de changement des pages et une stratégie de synchronisation (ou téléchargement) des pages.

De nombreuses expériences (Coffman et coll., 1997; Brewington et Cybenko, 2000; Cho et Garcia-Molina, 2003b) ont montré qu'une loi de Poisson modélisait correctement les mises à jour aléatoires et indépendantes des pages Web. Cet estimateur peut être basé sur plusieurs mesures de fraîcheur (Cho et Garcia-Molina, 2003a) : une mesure binaire (*freshness*) où la fraîcheur d'une page vaut 1 si elle est fraîche et 0 sinon. Et une mesure continue (*âge*) où la fraîcheur d'une page est négativement proportionnelle à la dernière date de visite. En marge de ces mesures, Pandey et Olston (2008) ont proposé un modèle directement fondé sur les modifications des pages et non sur l'âge de la page (*information longevity*).

Cho et Garcia-Molina (2003a) ont également proposé plusieurs modèles pour déterminer la stratégie de synchronisation optimale. Un premier modèle opère une mise à jour de toutes les pages avec une même fréquence alors qu'un second modèle opère cette mise à jour avec une fréquence de changement différente pour chaque page. La première stratégie a étonnamment été montrée comme supérieure. De plus, Cho et Garcia-Molina ont montré qu'il est plus avantageux d'exclure les pages qui sont mises à jour trop souvent afin de libérer de la bande passante pour mettre à jour plus de pages. Edwards et coll. (2001) ont, quant à eux, proposé de classer les pages en groupe de pages de fréquence de changement similaire et d'utiliser cette information pour améliorer la stratégie de re-*crawling*.

2.5 Synthèse

Dans ce chapitre, nous nous sommes intéressés à deux approches de constitution d'index spécialisés : l'utilisation d'un moteur de recherche Web et le *crawling* orienté.

La première approche a été relativement peu étudiée dans un but de création de moteurs de recherche spécialisés. Les requêtes formulées aux moteurs de recherche Web sont typiquement des combinaisons de termes du domaine fournis par un utilisateur ou extraits automatiquement à partir de petites collections documentaires existantes. Ces travaux se fondent sur l'hypothèse que la combinaison de mots-clés d'un domaine permet de désambiguïser le thème de la requête et d'obtenir une majorité de documents du thème. Toutefois, la validité de cette hypothèse n'a jamais été confirmée rigoureusement et dépend de plusieurs paramètres tels que le nombre de termes utilisés, le caractère discriminant des termes choisis, ou encore l'ambiguïté de la terminologie du domaine. C'est pourquoi une étape de validation manuelle des documents est généralement appliquée.

Les travaux en *crawling* orienté ont, eux, été beaucoup plus nombreux. Nous constatons que les travaux récents sur ce domaine font usage de méthodes d'apprentissage pour mieux ordonner la frontière de *crawl* ou tenter d'augmenter le rappel du *crawler*. Les méthodes utilisées dans ce domaine sont par ailleurs analogues à celles appliquées en recherche d'information : la requête utilisateur est remplacée par un thème, mais l'objectif du *crawler* orienté est bien de trouver l'ensemble de documents les plus pertinents possible pour le thème formulé.

Enfin, nous avons proposé une vue d'ensemble des méthodes de revisite. Bien que ces méthodes ne soient pas spécifiques aux moteurs de recherche spécialisés, la revisite est un élément clé de ces moteurs qui doivent offrir une fraîcheur exemplaire pour concurrencer les moteurs de recherche généralistes.

Deuxième partie

COLLECTE AUTOMATIQUE DE DOCUMENTS SPÉCIALISÉS

Nous avons vu, au travers des travaux présentés au chapitre précédent, que deux méthodes principales pour la collecte de documents Web sont : *(i)* utiliser un moteur de recherche comme point d'entrée au Web, et *(ii)* effectuer un parcours (*crawl*) orienté du Web.

Pour la première approche, nous avons constaté que les utilisateurs peuvent influer de deux manières sur la qualité des corpus produits :

- En amont du moteur de recherche, en formulant de bonnes requêtes, en choisissant, par exemple, des termes du domaine plus discriminants ;
- En aval du moteur de recherche, en filtrant et en retravaillant les documents renvoyés par le moteur de recherche.

Nous proposons, dans un premier temps, d'évaluer les performances de l'approche par combinaison de termes d'un domaine pour la constitution de corpus thématique à partir du Web (Chapitre 3). Cette première évaluation est un préalable nécessaire pour la suite de nos travaux. Elle nous permettra également de constater les performances de cette approche dans un cadre rigoureux, objectif et reproductible, ce qui n'avait pas été réalisé précédemment. Puis, au Chapitre 4, nous définissons une nouvelle mesure du caractère discriminant d'un terme pour un thème. Nous évaluons l'intérêt de cette mesure pour la sélection de bons termes amorces et de requêtes, une problématique cruciale pour la création de corpus thématiques. Enfin, nous présentons une méthode de collecte et de filtrage non supervisée fondée sur un modèle de graphe et un algorithme de marche aléatoire, au Chapitre 5.

Comme nous l'avons mentionné à la Section 2.3, l'utilisation d'un moteur de recherche Web comme source de documents impose plusieurs limites qui peuvent être outrepassées en réalisant son propre parcours thématique du Web. Comme nous l'avons vu au chapitre précédent, la littérature concernant le *crawling* orienté est relativement fournie, et deux problématiques en particulier semblent intéressantes pour améliorer l'état de l'art : *(i)* améliorer l'ordonnancement de la frontière de *crawl* au moyen d'algorithmes d'apprentissage permettant de prédire la présence de pages pertinentes proches ; *(ii)* prendre en compte l'effet tunnel pour augmenter la couverture du corpus. Nous explorons ces pistes dans le Chapitre 6. Nous discutons en premier lieu de l'implémentation de notre *crawler* orienté *Babouk*, puis nous proposons un nouveau modèle d'ordonnancement de la frontière de *crawl*. Ce modèle est basé sur un algorithme d'apprentissage de fonctions d'ordonnancement (*Learning to Rank*) entraîné sur des données de *crawls* existants et visant à ordonner la frontière de *crawl*.

3

CONSTRUCTION DE BASES DOCUMENTAIRES *VIA* UN MOTEUR DE RECHERCHE

> *“ The query is the loadstone of search, the runes we toss in our ongoing pursuit of the perfect results. ”*
>
> John Battelle, The Search, 2005.

Les moteurs de recherche Web actuels offrent bien peu de marge quant à la formulation du besoin informationnel de l'utilisateur. À titre d'exemple, Google offre les opérateurs suivants : “recherche d'expression”, -exclure, ~synonymes, opérateurs AND/OR booléens, caractère * générique. La pierre angulaire à l'obtention de pages pertinentes est donc la sélection des termes de la requête. Il n'est, par conséquent, pas surprenant que les travaux présentés en Section 2.2 s'appuient unanimement sur des combinaisons de termes appartenant à un domaine pour trouver des documents appartenant à ce même domaine.

La sélection de ces termes amorces n'est pourtant pas une tâche aisée. Elle se fonde souvent sur des mesures subjectives de la « spécialisation » du terme ou de son caractère discriminant pour le thème. De plus, certains thèmes (biologie, cuisine) possèdent une terminologie claire et peu ambiguë comparativement à d'autres (sociologie), qui partagent une grande partie de leur vocabulaire avec la langue commune (Kluck et Gey, 2001).

Il nous semble dès lors pertinent de nous interroger quant aux véritables performances que fournit une approche par combinaison de termes sur un large ensemble de thèmes. En effet, peu de travaux, à notre connaissance se sont intéressés à une évaluation systématique, objective et reproductible, des performances de telles approches. Nous proposons dans ce chapitre une méthode d'évaluation et son application pour évaluer la pertinence des bases documentaires obtenues par ce procédé, et donc la quantité de filtrage nécessaire a posteriori.

Le Web est un univers changeant (ajouts, suppressions, modifications des pages), tout comme les moteurs de recherche Web dont les index et les fonctions d'ordonnancement évoluent quotidiennement. Pour proposer une évaluation rigoureuse, objective et reproductible, nous devons nous

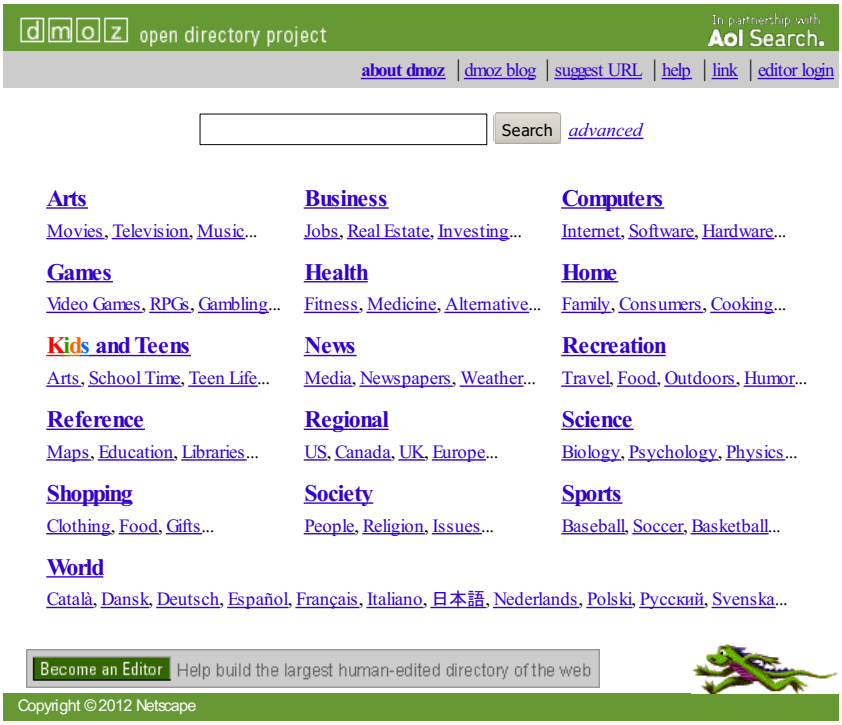


FIGURE 3.1 : Page d'accueil du site de l'OpenDirectory au 11 décembre 2012.

baser sur une collection documentaire annotée et stable, tout en étant relativement représentative du Web. Pour ce faire, nous nous appuyons sur un large corpus de pages Web annoté thématiquement : l'OpenDirectory (Figure 3.1). Après avoir indexé ce corpus dans un moteur de recherche, nous allons créer des listes de termes pour les différents thèmes du corpus. Puis, nous allons générer des combinaisons de ces termes et évaluer le nombre de documents pertinents renvoyés pour chaque requête.

3.1 Données

L'OpenDirectory, parfois nommé ODP ou DMOZ (pour *Directory Mozilla*), est un répertoire de sites Web maintenu par une communauté d'éditeurs bénévoles. Le répertoire comprend en Décembre 2012 un peu plus de 5 millions d'entrées réparties sur 78 langues. Les sites sont ordonnés dans un thésaurus hiérarchique et chaque entrée est annotée par une courte description de son contenu. Les données de l'OpenDirectory sont mises à jour régulièrement et sont disponibles au téléchargement au format RDF (XML). Quatre exemples d'entrées, trois en français et un en anglais, sont présentés au Tableau 3.1.

URL	http://www.cours-medecine.info/anatomie/
Titre	Cours d'Anatomie
Catégorie	`Santé/Médecine/Anatomie`
Description	Présente des cours d'anatomie humaine, des fiches de résumés, schémas et coupes à légender, ainsi que des QCM.

http://trec.nist.gov/
Text REtrieval Conference
`Computers/Software/Information_Retrieval`
An annual information retrieval conference and competition, the purpose of which is to support and further research within the information retrieval community.

http://www.geomagazine.fr/
GEO
`Loisirs/Voyage/Publications`
Magazine papier mensuel de reportages photographiques des quatre coins du monde. Sommaire et édito du numéro en kiosque, abonnement, index de tous les articles publiés.

http://www.cahiersducinema.com/
Les cahiers du cinéma
`Arts/Audiovisuel/Cinéma/Actualité_et_médias`
Version en ligne de la revue. Actualité, agenda, sorties et articles.

Tableau 3.1 : Exemples d'entrées issues de l'OpenDirectory.

Pour cette expérience, nous avons utilisé une version du répertoire datant du 15 septembre 2011 qui n'inclut pas les catégories `Adult` et `Kids_and_Teens` [1]. Nous nous sommes limités à la partie anglaise du répertoire, c'est-à-dire tout le répertoire excepté les entrées sous la catégorie `World`.

L'OpenDirectory ne fournit pas le contenu des pages Web, mais uniquement leur adresse URL. Par conséquent, il est nécessaire de les télécharger en sus du répertoire de l'OpenDirectory. Le téléchargement de plusieurs millions de pages ainsi que leur traitement nécessite une certaine technicité (gestion des temps d'attente, des redirections, des différents encodages, ...) qui, une fois en place, nous a permis de récupérer 2 339 125 pages sur un total de 2 463 769 URL en anglais. Le faible pourcentage de pages

1. Ces deux catégories sont distribuées dans des archives séparées. `Kids_and_Teens` est un répertoire pour les personnes de moins de 18 ans (http://www.dmoz.org/guidelines/kguidelines/) et `Adult` contient uniquement des documents à caractère sexuel (http://www.dmoz.org/guidelines/adult/).

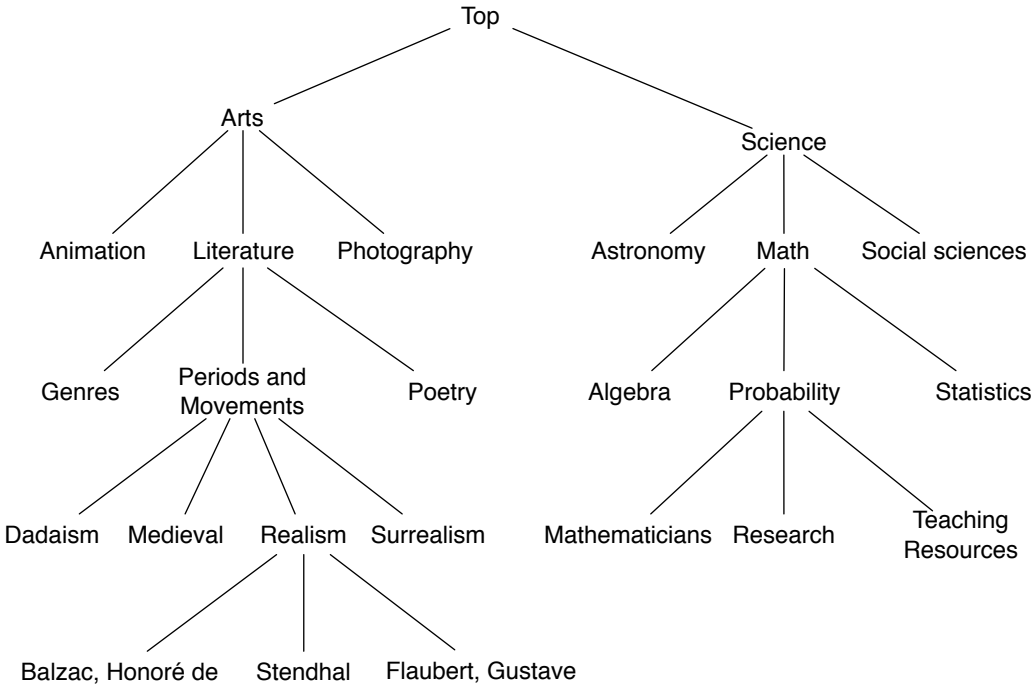


FIGURE 3.2 : Extrait de l'arborescence de catégories de l'OpenDirectory.

perdu est réparti sur l'ensemble des catégories et n'a donc, statistiquement parlant, pas d'impact sur nos conclusions.

Une étude préliminaire du répertoire a montré qu'une partie des catégories de l'OpenDirectory n'avaient pas de caractère thématique. Nous appliquons donc une phase supplémentaire de filtrage dans le but d'améliorer la qualité du corpus. Nous avons exclu toutes les entrées sous la catégorie `Regional` qui contient une classification géographique, et non thématique, des sites Web [2]. Un certain nombre de catégories ne correspond pas à des thèmes, mais à des genres de pages (Mehler et coll., 2010) et nous supprimons donc les catégories `Chats_and_Forums`, `Directories`, `FAQs,_Help,_and_Tutorials`, `Magazines_and_E-zines`, `Mailing_Lists`, `News`, `News_and_Media`, `Personal_Pages`, `Search_Engines`, et `Weblogs`, ainsi que les catégories purement organisationnelles (`By_Culture`, `By_Region`, `By_Type`, ...), les catégories « lettres » (listes de sportifs dont le nom commence par la lettre X) et les catégories « dates » (`Roland_Garros/2005,2006,2007,...`). Enfin, dans chacune de nos évaluations, nous ignorons les catégories contenant moins de 50 documents.

2. http://www.dmoz.org/docs/fr/guidelines/subcategories.html#regional

Profondeur	Nb catégories	Nb documents	Nb documents/catégorie
1	13	1 261 095	97 007.3
2	340	1 261 090	3 709.1
3	4 583	1 257 592	274.4
4	19 853	1 166 199	58.7
5	45 905	896 534	19.5

Tableau 3.2 : Nombre de documents et catégories en fonction de la profondeur de l'arbre de catégories de l'OpenDirectory. Seules les valeurs pour les 5 premiers niveaux de l'arbre sont présentées.

Nous avons ensuite indexé les entrées de l'OpenDirectory dans *Lucene*[3], un moteur de recherche open source basé sur le modèle vectoriel[4] (Salton et Buckley, 1988). Un nettoyage rudimentaire est appliqué aux pages Web : les scripts dynamiques, feuilles de styles et autres balises HTML sont supprimés. Le texte est ensuite segmenté (*tokenisé*) et une racinisation (algorithme de Porter (1980)), disponible par défaut dans *Lucene*, est appliquée. Nous stockons également, pour chaque document, sa ou ses catégorie(s), sa description, son URL, ainsi que son titre. Le corpus final, une fois téléchargé, filtré et indexé, comporte un ensemble de 171 954 catégories et 1 261 095 documents au total. L'arbre de catégories possède une profondeur moyenne de 6 pour une profondeur maximale de 12. Seuls 3% des documents sont *multilabels*, c'est-à-dire qu'ils appartiennent à plus d'une catégorie. Un extrait de l'arbre de catégories est présenté à la Figure 3.2.

Nous présentons au Tableau 3.2 quelques caractéristiques des 5 premiers niveaux de l'arbre de catégories. Nous constatons que le nombre moyen de documents par catégories chute fortement à chaque niveau. À partir d'une profondeur de 4, le nombre de documents par catégorie devient trop faible pour mener à bien notre évaluation.

Pour notre évaluation, nous sélectionnons arbitrairement toutes les catégories du niveau 2 de l'arbre. Nous justifions ce choix par le fait que les catégories à ce niveau correspondent à nos attentes en terme de granularité de thème et suivent la logique des moteurs de recherche thématiques existants présentés précédemment au Tableau 1.1 (§ 1.4, p. 13). Quelques exemples de catégories au second niveau de l'arbre de catégories sont : `Arts/Design`, `Business/Energy`, `Computers/Home_Automation`,

3. http://lucene.apache.org/
4. http://lucene.apache.org/core/old_versioned_docs/versions/2_9_3/scoring.html

`Science/Chemistry`, `Society/Politics` et `Sports/Hockey`.

3.2 Sélection des termes amorces

Nous nous intéressons à présent à la sélection des termes amorces qui vont servir à créer les requêtes spécialisées. Nous nous appuyons sur les descriptions des sites internet entrées manuellement par les éditeurs de l'OpenDirectory. Pour chaque catégorie, nous agrégeons les descriptions des sites qui la composent en un seul texte afin de générer une *description de catégorie* (ou *topic description* (Srinivasan et coll., 2005)). Nous modélisons ensuite chaque description de catégorie en un sac de mots et appliquons une mesure de Spécificité (*TermHood* (Kageura et Umino, 1996)), en l'occurrence, le *tfidf* (Salton et Buckley, 1988), pour trouver les termes les plus importants pour cette catégorie. La mesure du *tfidf* est elle-même composée de deux mesures : le *tf*, pour *term frequency*, qui mesure l'importance locale d'un terme dans un document (ici, une catégorie), et le *df*, pour *document frequency*, qui mesure l'importance globale d'un terme dans une collection (ici, l'ensemble des catégories).

Nous utilisons la variante *ntc* (Manning et coll., 2008, chap. 6, p. 128) du tfidf qui est définie comme suit :

$$\text{tfidf}_{t,d} = \text{tf}_{t,d} \times \log \text{idf}_t \qquad \text{idf}_t = \frac{N}{\text{df}_t}$$

avec $\text{tf}_{t,d}$ le nombre d'occurrences du terme t dans le document d, df_t le nombre de documents où apparaît le terme t et N le nombre total de documents dans la collection.

Nous présentons au Tableau 3.3 les 10 meilleurs termes obtenus *via* cette mesure pour quelques catégories choisies manuellement.

3.3 Protocole expérimental

Pour évaluer la qualité des corpus produits en sortie de l'approche par combinaison de termes, nous appliquons la procédure suivante pour chaque catégorie :

1. Nous sélectionnons les N meilleurs termes pour une catégorie en suivant la méthode décrite à la section précédente ;
2. Nous construisons des combinaisons de ces termes (ou tuples) de différentes tailles K, allant de 1 à 3 ;

Business/Energy	Computer/A.I.	Science/Math	Society/Paranormal
solar	neural	mathematics	psychic
gas	learning	mathematical	readings
energy	algorithms	algebraic	paranormal
oil	reasoning	algebra	clairvoyant
electric	networks	theory	tarot
water	machine	geometry	ufo
biodiesel	bayesian	math	ghost
electricity	ai	department	hauntings
drilling	computational	equations	intuitive
wind	intelligence	logic	ghosts

Tableau 3.3 : Dix termes ayant le plus fort *tfidf* pour quatre catégories du second niveau de l'OpenDirectory.

3. Nous générons des requêtes en liant les tuples avec un opérateur de conjonction (*AND*) ;
4. Nous soumettons les requêtes au moteur de recherche (*Lucene*) et récupérons les (au plus) M meilleurs résultats ;
5. Nous analysons la qualité du corpus produit en étudiant les catégories des documents renvoyés.

Pour analyser la qualité de l'ensemble de documents ramenés pour une catégorie, nous nous appuyons sur les mesures de précision et de rappel qui permettent d'évaluer une classification binaire (ici, appartenant au thème ou non). Ces deux quantités sont définies pour évaluer des ensembles de résultats, sans notion d'ordre. Elles ont été développées dans le cadre de la recherche d'information et sont définies comme suit (Manning et coll., 2008, chap. 8, p. 155) :

$$\text{Précision} = \frac{\#(\text{éléments pertinents ramenés})}{\#(\text{élément ramenés})}$$

$$\text{Rappel} = \frac{\#(\text{éléments pertinents ramenés})}{\#(\text{élément pertinents})}$$

Dans un cadre de classification, ces deux mesures sont généralement interprétées à l'aide d'une table de contingence, également appelée matrice de confusion, présentée au Tableau 3.4.

		Catégorie Réelle	
		Positif	Négatif
Catégorie Prédite	Positif	VP Vrai Positif	FP Faux Positif
	Négatif	FN Faux Négatif	VN Vrai Négatif

Tableau 3.4 : Matrice de confusion définie à partir de données annotées (catégories *réelles*) et de catégories prédites

Dans ce cadre, la précision et le rappel sont alors définis comme :

$$\text{Précision} = \frac{\text{VP}}{\text{VP} + \text{FP}} \qquad \text{Rappel} = \frac{\text{VP}}{\text{VP} + \text{FN}}$$

Ces deux mesures sont généralement liées, et il est dès lors possible d'accroître l'une d'elles au détriment de l'autre. Une précision parfaite peut par exemple être obtenue en ne classant que peu de documents pour lesquels le classifieur a une confiance élevée. Au contraire, il est possible d'obtenir un rappel élevé en classant tous les documents dans la catégorie étudiée. Le choix de favoriser l'une ou l'autre des mesures dépend alors de l'application considérée.

Pour faciliter l'évaluation d'un classifieur, une unique mesure, moyenne harmonique de la précision et du rappel, a été définie par Rijsbergen (1979) : la F_β-Mesure. Le paramètre β équilibre alors la précision et le rappel. La F_1-Mesure ($\beta = 1$), qui pondère équitablement précision et rappel, est communément utilisée et est définie comme suit :

$$F_1\text{-Mesure} = 2 \cdot \frac{\text{Précision} \cdot \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

Nous nous sommes pour l'instant concentrés sur l'évaluation indépendante de chaque catégorie. Pour évaluer les performances de l'approche sur un ensemble de catégories, il convient de calculer une moyenne des performances obtenues pour chaque catégorie. Cette moyenne peut être calculée de deux manières distinctes :

- Macro moyenne : les mesures sont effectuées pour chaque catégorie puis les résultats sont moyennés sur l'ensemble des catégories.
- Micro moyenne : les mesures sont effectuées sur l'ensemble des documents.

La différence entre ces deux moyennes réside dans le fait que la macro moyenne pondère les différentes catégories équitablement, alors que

la micro moyenne pondère les documents équitablement et favorise ainsi les catégories possédant un plus grand nombre de documents. Nous utiliserons ces deux moyennes en fonction de l'équilibre des catégories des données étudiées et des objectifs visés.

3.4 Évaluations

Pour ces évaluations, nous utilisons les paramètres suivants, sauf mention explicite : nous fixons le nombre de termes amorces sélectionnés par catégorie (N) à 10, tout comme le nombre de documents à télécharger par requête (M)[5]. Dans un premier temps, nous faisons varier la taille des tuples (K) entre 1 et 4, puis nous la fixons à 3.

3.4.1 *Performances en fonction de la taille des tuples*

Nous étudions tout d'abord l'influence de la taille des tuples sur la qualité des documents collectés. Pour ce faire, nous réalisons, pour chaque catégorie, toutes les combinaisons possibles à partir des 10 termes ayant le plus fort *tfidf*. Puis nous mesurons la précision (à 10) et le rappel des documents renvoyés pour chaque requête. Nous considérons qu'un document est pertinent si sa catégorie dans l'OpenDirectory est la même que celle qui a servi à générer les termes de la requête. Nous calculons enfin une micro moyenne de ces résultats.

Nous présentons au Tableau 3.5 les micro moyennes des précisions et rappels obtenues pour l'ensemble des requêtes. Nous observons clairement un écart important entre les requêtes composées d'un seul terme, trop ambiguës, et les requêtes composées de 2 termes et plus. Nous constatons que la précision augmente avec le nombre de termes par requête : l'ajout de termes contraint l'ensemble de documents en sortie et valide le thème des documents. Cependant, le gain obtenu en augmentant la taille des requêtes décroît rapidement ($+0,001$ en précision entre des requêtes de 3 et 4 termes). Ce dernier résultat semble indiquer que l'augmentation de la taille des requêtes ne permet pas d'améliorer la précision substantiellement au-delà de quelques termes.

Force est de constater qu'une valeur de précision de $0,46$ est relativement faible. Nous pensons que les performances que nous obtenons représentent une borne inférieure aux performances que nous pouvons obtenir avec un moteur de recherche Web. En effet, d'une part le modèle vectoriel *tfidf* employé par Lucene est inférieur à l'état de l'art actuel (He et Ounis,

5. Ce choix est simplement motivé par le fait que cela correspond généralement au nombre d'entrées présentées sur la première page de résultats d'un moteur de recherche Web.

Taille	Nb requêtes	Précision	Rappel
1	10	0,228	0,003
2	45	0,404	0,006
3	120	0,457	**0,007**
4	210	**0,458**	0,006

Tableau 3.5 : Micro moyennes de la précision et du rappel pour des requêtes de tailles différentes.

Taille	Nb requêtes	Nb documents	Précision	Rappel	F_1-Mesure
1	10	87,9	0,216	0,025	0,045
2	45	245,6	0,329	0,084	0,134
3	120	321,9	0,353	**0,101**	**0,157**
4	210	297,4	**0,373**	0,094	0,151

Tableau 3.6 : Macro moyenne de la précision, du rappel et de la F_1-mesure incluant le recouvrement entre requêtes.

2005). D'autre part, les moteurs de recherche Web font usages de nombreuses données (traces des utilisateurs par exemple) qui leurs permettent d'améliorer leur fonction d'ordonnancement. En l'état, ce résultat laisse une marge d'amélioration intéressante. L'application d'un filtre *a posteriori*, manuelle ou automatique, semble inévitable.

Les mesures que nous avons présentées ne sont pas suffisantes pour sélectionner une taille de requête optimale. Ces résultats ne tiennent pas compte du recouvrement entre les différentes requêtes d'une même catégorie. En effet, nous nous attendons à un recouvrement plus fort entre requêtes de grande taille (4 termes par exemple) que pour des requêtes de taille inférieure. Nous étudions dès lors la précision et le rappel, non plus pour chaque requête, mais pour l'ensemble des requêtes d'une catégorie. Notons qu'il ne s'agit pas simplement de la macro moyenne des mesures mentionnées précédemment, mais de mesures différentes en raison du recouvrement entre les documents ramenés par les requêtes d'une même catégorie. Comme précédemment, nous partons des 10 meilleurs termes pour une catégorie et générons toutes les combinaisons de ces termes de manière exhaustive. Puis, nous évaluons la précision et le rappel de l'union des documents ramenés par toutes les requêtes d'une catégorie.

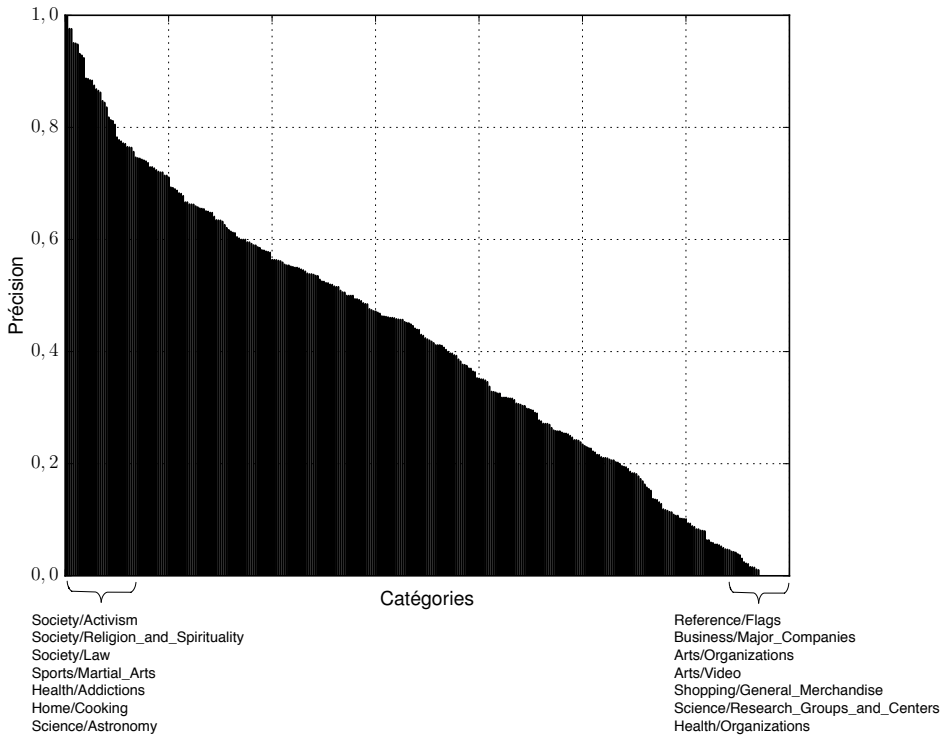


FIGURE 3.3 : Précision pour chaque catégorie avec des requêtes de taille 3.

Nous observons au Tableau 3.6 que le rappel atteint une valeur maximale pour des requêtes composées de 3 termes. Au-delà de ce seuil, le rappel et le nombre de documents uniques diminuent, alors que la précision continue d'augmenter. En effet, l'ajout de termes dans les requêtes contraint d'autant plus l'ensemble de documents en sortie, ce qui permet d'améliorer la qualité du corpus (augmentation de la précision) mais diminue la couverture de ce corpus (diminution du rappel). Nous constatons également que le rappel est très faible (inférieur ou égal à 0,1). Il est d'une part logique que le rappel soit faible, car nous formulons entre 10 et 210 requêtes et ne téléchargeons que 10 documents par requêtes. De plus, considérant que les requêtes font intervenir plusieurs fois les mêmes termes, nous pouvons attendre un recouvrement fort entre les documents renvoyés, et donc une faible fraction des documents pertinents.

Dans notre cas, nous visons une précision maximale et un rappel raisonnable : en effet, nous voulons conserver la possibilité d'augmenter notre corpus en proposant de nouveaux termes et en formulant de nouvelles requêtes. Cependant, nous ne souhaitons pas non plus un rappel trop faible qui nous obligerait à formuler un nombre de requêtes important et donc coûteux. La F_1-mesure, qui pondère équitablement précision et rappel, est

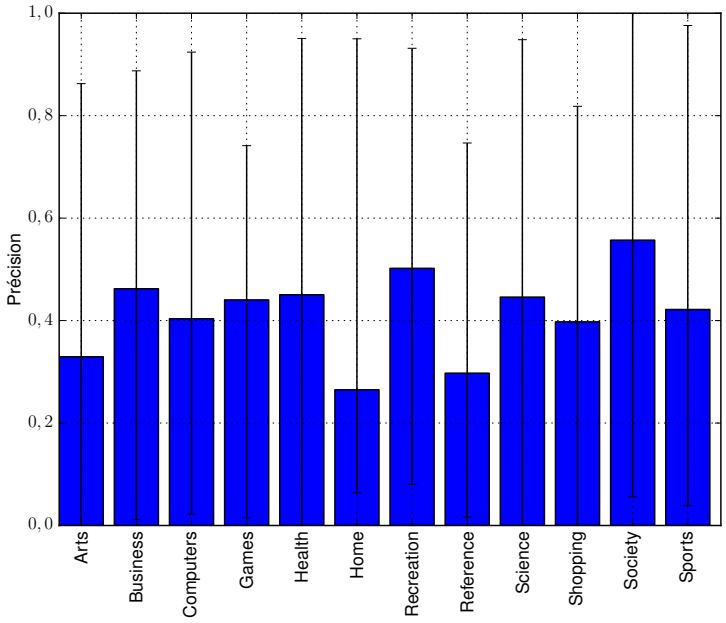


FIGURE 3.4 : Macro moyenne par catégorie de premier niveau de la précision pour les catégories du second niveau

également maximale pour des requêtes de 3 termes. Nous optons donc pour des requêtes de cette taille pour le reste de nos expériences.

3.4.2 *Performances en fonction des catégories*

Nous profitons de ce corpus pour évaluer la difficulté de collecte en fonction des catégories. Nous mesurons la précision des collections créées pour chaque catégorie.

Le résultat, présenté à la Figure 3.3, ne fait pas apparaître de groupes de catégories plus simples ou plus difficiles à construire. De même, la Figure 3.4 montre la macro moyenne par catégorie de premier niveau de la précision des catégories de second niveau. Nous constatons que la précision sur ces catégories peut varier de plus ou moins $0,2$. Toutefois, les catégories de second niveau à l'intérieur de ces catégories ont une précision qui varie également beaucoup (symbolisée par les barres d'erreurs). Une étude plus approfondie des résultats montre que les mauvaises performances obtenues sur certaines de ces catégories peuvent être dues à plusieurs facteurs, tels que :

1. Des mot-clés mal choisis (*tfidf*) ;
2. Des catégories similaires existantes (`Arts/Video`, `Arts/Movies/Filmmaking` et `Arts/Movies/Education`, par exemple).

3.5 Bilan

Dans ce chapitre, nous nous sommes intéressé à une approche standard de recherche orientée. Dans ce cadre, nous avons proposé une méthode d'évaluation rigoureuse, claire et facilement reproductible pour mesurer les performances de ces systèmes de collecte. Cette méthode se fonde sur l'indexation d'un sous-ensemble annoté du Web (l'OpenDirectory) dans un moteur de recherche (*Lucene*). Nous avons ensuite appliqué notre méthode pour évaluer les performances d'une approche simple pour la collecte de documents thématiques *via* un moteur de recherche : l'approche par combinaison de termes. Nous avons observé des résultats variables pour les différentes catégories, sans pour autant détecter de tendance claire quant à la difficulté de groupes thématiques de catégories. Nous avons constaté que la création de triplets de termes offre la meilleure F_1-mesure sur nos données. Cependant, même la précision maximale, obtenue pour des requêtes composées de quatre termes, reste moyenne $(0,373)$ et le gain pouvant être obtenu en augmentant la taille des requêtes semble limité. Ces résultats laissent la porte ouverte à d'intéressantes améliorations au niveau de la formulation du besoin thématique (termes composant la requête) mais également du filtrage des documents rapatriés.

4 PRÉDICTION DES PERFORMANCES DES REQUÊTES THÉMATIQUES SUR LE WEB

MOTIVATION

Comme nous l'avons mentionné en introduction de cette partie (p. 33), la qualité de la collecte de documents thématiques *via* un moteur de recherche peut être ajustée en amont et en aval du moteur de recherche. En amont du moteur, l'objectif est de fournir de bonnes requêtes au système de collecte, ce qui signifie principalement de bons termes. Or, nous avons constaté au chapitre précédent que la sélection de termes amorces *via* la mesure du *tfidf* fournissait des performances mitigées et variables (Tableau 3.6).

Nous nous intéressons dès lors à la définition d'une mesure de prédiction des performances des requêtes thématiques sur le Web. Il est important que cette mesure soit représentative du Web qui introduit des ambiguïtés que l'on ne retrouve pas nécessairement dans des corpus généraux. En outre, il semble intéressant, bien qu'optionnel, que cette mesure soit calculée spécifiquement pour le moteur de recherche utilisé par la suite pour collecter les corpus thématiques.

Nous envisageons plusieurs applications pour cette mesure : *(i)* elle peut permettre à un utilisateur d'estimer et d'ajuster sa sélection de termes et de requêtes ; *(ii)* elle peut permettre de favoriser certaines requêtes lors d'une sélection aléatoire parmi un large ensemble de requêtes ; *(iii)* elle peut permettre d'orienter le filtrage vers les documents récupérés par des requêtes faiblement pertinentes.

MÉTHODE

Pour calculer la performance d'une requête thématique, nous proposons dans un premier temps de calculer la performance de chaque terme composant cette dernière de manière indépendante. Puis, nous calculons la performance d'une requête thématique comme la « somme » des performances de chaque terme la composant. Cette approximation nous permet de calculer un score pour chaque combinaison de termes sans pour autant avoir à soumettre toutes ces combinaisons à un moteur de recherche.

En entrée de notre procédure, nous disposons d'un ensemble de termes liés thématiquement, que nous nommons *lexique thématique*, dont nous souhaitons calculer la performance. Ces termes peuvent être choisis manuellement ou automatiquement à partir de documents existants. Si l'utilisateur dispose d'un ensemble de documents, les termes simples et/ou complexes sont extraits et pondérés *via* une première mesure de sélection (*tfidf*, *tf*)[1]. Les N termes les mieux classés sont ensuite extraits et utilisés comme lexique thématique. Si l'utilisateur dispose d'un ensemble d'URL, les pages Web sont téléchargées et nettoyées de leurs balises. Puis, elles sont traitées comme un ensemble de documents.

Partant d'un lexique thématique, nous calculons notre mesure de performance pour chaque terme en deux étapes. Tout d'abord, nous recueillons des connaissances exogènes sur chaque terme du lexique thématique. Pour ce faire, nous utilisons notre moteur de recherche et collectons un ensemble de documents contenant chaque terme. Notons qu'il est également possible d'approximer cet ensemble de documents à l'aide d'un autre moteur de recherche Web, ou même un moteur de recherche local indexant un précédent *crawl* du Web tel que le corpus *ClueWeb2009*[2] ou le corpus *commoncrawl*[3]. Une fois ces documents collectés, nous appliquons un critère original, dénommé critère de *cohésion thématique*, qui s'appuie sur les cooccurrences des termes du lexique dans les documents collectés pour estimer la pertinence de chaque terme pour le thème.

4.1 Travaux liés

La prédiction des performances d'une requête a pour objectif d'estimer la qualité des résultats renvoyés par un moteur de recherche pour cette requête. Les applications en recherche d'information sont nombreuses et incluent, par exemple, l'application d'une reformulation de la requête ou d'une correction orthographique, afin d'améliorer la qualité des résultats présentés à l'utilisateur.

Les approches en prédiction des performances d'une requête s'inscrivent dans deux courants (Hauff, 2010) : *(i)* utiliser des informations disponibles avant la recherche (*pre-retrieval*), c'est-à-dire issues de la collection documentaire ; *(ii)* utiliser des informations disponibles après la recherche (*post-retrieval*), c'est-à-dire issues de l'ensemble de documents renvoyé par le

1. Après avoir extrait un ensemble de termes candidats à l'aide de patrons morpho-syntaxiques, des critères de sélection simples tels que le *tf* fournissent de bons résultats (Wermter et Hahn, 2006; de Groc, 2010).
2. http://www.lemurproject.org/clueweb09.php/
3. http://commoncrawl.org/

moteur de recherche.

L'évaluation des méthodes de prédiction est généralement basée sur un score de corrélation entre le score prédit et l'efficacité réelle de la requête, mesurée à l'aide d'annotations et de mesures telles que la précision et la précision moyenne (Hauff, 2010). Si la valeur de la prédiction importe, alors le coefficient de corrélation linéaire (également nommé « r de Pearson ») est utilisé. Si l'objectif est plutôt d'ordonner un ensemble de requêtes en fonction de leurs scores, par exemple pour prédire la meilleure reformulation pour une requête, alors une mesure de corrélation de rangs est utilisée (ρ de Spearman ou τ de Kendall).

Dans le cadre de nos travaux, nous disposons uniquement d'un ensemble de termes liés thématiquement en entrée. Nous nous intéressons donc plus particulièrement aux approches dites *post-retrieval*. Dans ce contexte, l'approche la plus efficace et la plus populaire est la mesure de clarté d'une requête (*clarity*) définie par Cronen-Townsend et coll. (2002). Cette mesure est basée sur la notion de modèle de langue et est définie comme la distance de Kullback-Leibler entre le modèle de langue de la requête et celui de la collection documentaire du moteur de recherche. Les auteurs ont montré une corrélation positive entre la mesure de clarté et la précision moyenne des requêtes sur leur collection.

He et coll. (2008) s'appuient sur une mesure de *cohérence de la requête* (*QC-2*) qui évalue conjointement :

1. la cohérence moyenne de l'ensemble des documents renvoyés pour chaque terme de la requête ;
2. la cohérence globale entre les ensembles de documents renvoyés pour chacun des termes de la requête.

Les auteurs montrent une corrélation positive entre leur mesure et la précision moyenne.

Jensen et coll. (2005) proposent une méthode de prédiction de la difficulté d'une requête fondée sur des traits « visuels » issus des pages de résultats des moteurs de recherche. Ces pages fournissent les titres, les URL et de courts extraits (ou *snippets*) des documents pour un coût très faible en comparaison du téléchargement de ces mêmes documents. Les auteurs s'appuient sur 31 traits extraits de ces pages de résultats pour entraîner un algorithme de régression. Ces traits incluent par exemple le pourcentage de n-grammes de caractères ou de mots en commun entre la requête et le titre d'un résultat, son *snippet* ou son URL.

L'idée développée par Song et coll. (2007) est que les requêtes ambiguës mènent à des documents appartenant à des catégories thématiques différentes (*Computing*, *Library*, *Word & Money*). Ainsi, ils utilisent cette information dans un classifieur et démontrent un fort pourcentage d'exactitude quant à la prédiction de requêtes difficiles. Une idée similaire est développée par Qiu et coll. (2007), qui proposent d'exploiter l'OpenDirectory afin de mesurer l'ambiguïté d'une requête. Dans un premier temps, ils soumettent chaque terme de la requête au moteur de recherche de l'OpenDirectory [4], qui renvoie une liste de catégories correspondantes pour chacun de ces termes. Puis ils calculent une mesure d'ambiguïté en se basant sur l'intersection de ces catégories.

Nous allons voir, dans la section suivante, que notre approche est conceptuellement liée aux travaux de He et coll. (2008), car nous calculons un critère de cohérence *thématique* entre les termes de notre lexique. De plus, nous faisons une hypothèse proche de Song et coll. (2007) qui stipule que la pertinence d'un terme pour un thème est fonction du nombre de documents ramenés par le terme et appartenant effectivement au thème. Enfin, comme Jensen et coll. (2005), nous allons nous intéressés à l'utilisation des *snippets* des moteurs de recherche qui représentent une source d'information contextuelle précise à faible coût.

4.2 Pertinence d'un terme pour un thème

Nous formulons notre problématique comme suit : considérant un *lexique thématique* $\mathcal{L}_T$ composé de N termes, $\mathcal{L}_T = (t_1, t_2, \ldots, t_N)$, nous voulons calculer un vecteur de poids $\mathbf{w}_{\mathcal{L}_T} = (w_1, w_2, \ldots, w_N)$ où chaque poids w_i mesure la pertinence du terme t_i pour le thème T.

4.2.1 *Recueil de connaissances exogènes*

Nous collectons, pour chaque terme t_i, un corpus C_i correspondant aux M meilleurs résultats renvoyés par un moteur de recherche pour la requête ne contenant que ce mot ("$< t_i >$"). La valeur de M doit être suffisamment grande pour réduire le biais du moteur de recherche, tout en restant raisonnable afin d'éviter un temps de traitement trop long.

Nous considérons deux unités d'information : la page Web et le *snippet*. Prendre en compte les pages Web entières permet de bénéficier d'un contexte plus large et plus riche. Toutefois, télécharger les documents ren-

4. Ce moteur de recherche indexe uniquement les métadonnées présentes dans l'OpenDirectory.

voyés par le moteur de recherche rallonge considérablement le temps de calcul et nécessite une méthode de nettoyage des pages Web (suppression des menus, publicités et balises HTML). D'autre part, le caractère local des *snippets* peut permettre de réduire le bruit pouvant apparaître dans les pages Web.

4.2.2 *Critère de cohésion thématique*

Considérant un ensemble de documents (virtuellement infini) contenant le terme t_i, nous faisons l'hypothèse que la pertinence du terme t_i pour le thème T est fonction de la proportion de documents appartenant au thème T. Ainsi, le terme « *mercury* », par exemple, qui peut faire référence à un astre, à un dieu de la mythologie romaine, à un élément chimique ou à un célèbre chanteur, est relativement ambigu et devrait être discriminé. Au contraire, le terme « *telescope* », peu ambigu et menant très majoritairement à des documents traitant d'astronomie devrait obtenir un score important pour ce thème.

Cependant, nous ne disposons pas d'information quant au caractère thématique des documents. Une solution serait d'utiliser un catégoriseur thématique, à l'instar de Song et coll. (2007), mais là encore nous ne disposons pas d'un tel outil. Nous proposons plutôt d'estimer le caractère thématique des documents en fonction de l'apparition des termes du lexique dans le document. Par conséquent, nous proposons une première définition de notre critère comme suit : le poids w_i d'un terme t_i est égal au nombre de termes du lexique (t_i exclu) cooccurrant avec t_i dans le corpus C_i. Plus formellement, le poids w_i d'un terme t_i est défini par :

$$w_i = \sum_{t_j \in \mathcal{L}_T^\dagger} n_{t_j, C_i} \tag{4.1}$$

où n_{t_j,C_i} est le nombre d'occurrences du terme t_j dans l'ensemble des documents du corpus C_i et $\mathcal{L}_T^\dagger = \mathcal{L}_T \setminus \{t_i\}$ est l'ensemble des termes du lexique $\mathcal{L}_T$, t_i exclu.

Pour obtenir un score entre 0 et 1, nous modifions légèrement cette définition et ajoutons un facteur de normalisation :

$$w_i = \frac{\sum_{t_j \in \mathcal{L}_T^\dagger} n_{t_j, C_i}}{\sum_{k \in \{1,\ldots,|\mathcal{L}_T|\}} \sum_{t_j \in \mathcal{L}_T^\dagger} n_{t_j, C_k}} \tag{4.2}$$

Une seconde manière de voir ce critère est de représenter les cooccurrences des termes sous la forme d'un graphe orienté d'après le procédé suivant (également présenté à la Figure 4.1) : Soit $G = < V, E >$ un graphe

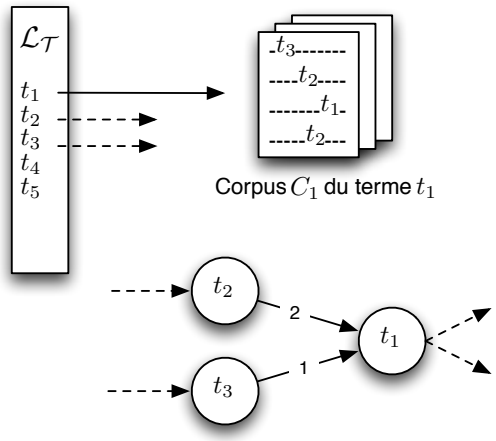


FIGURE 4.1 : Exemple de sous-graphe construit à partir du lexique $\mathcal{L}_\mathcal{T}$

orienté où V est l'ensemble des sommets ($V = \mathcal{L}_T$) et E l'ensemble des arcs. Chaque arc $e(t_i, t_j) \in E$ symbolise l'apparition du terme t_i dans le corpus C_j de t_j. Les arcs sont pondérés en fonction du nombre d'occurrences du terme t_i dans C_j. Le poids d'un terme tel que défini par l'équation 4.1 est alors équivalent au degré entrant pondéré d'un terme, c'est-à-dire la somme des poids des arcs entrants (Newman, 2004). De même, l'équation 4.2 est alors une version normalisée du degré entrant pondéré.

Notons que notre approche fondée sur un graphe orienté nous différencie des travaux de Mihalcea et Tarau (2004) qui extraient des termes saillants d'un texte en modélisant les cooccurrences de termes dans une fenêtre de taille fixe à l'aide d'un graphe non orienté. En effet, dans notre cas, pour deux termes t_i et t_j et leurs corpus associés C_i et C_j, l'apparition du terme t_i dans le corpus C_j constitue un indice du *vote* de t_i pour t_j. Cependant, cette relation n'est pas symétrique puisque les corpus C_i et C_j sont distincts.

Nous présentons à la Figure 4.2, un graphe « jouet » construit à partir de quelques noms de fruits. La direction des arcs est donnée par le sens de rotation des aiguilles d'une montre (Fekete et coll., 2003). L'intensité de la couleur des termes reflète la valeur du degré entrant pondéré normalisé (équation 4.2). Nous constatons que le terme « Apple », qui désigne à la fois le fruit et l'entreprise de Steve Jobs, possède peu d'arcs entrants et sera donc correctement pénalisé. Enfin, le terme « Mandarine » renvoie des documents contenant fréquemment le terme « Lemon » ce qui lui confère un poids important.

Le critère défini à l'équation 4.2 considère que tous les termes du lexique t_j ont une influence équivalente sur le poids du terme t_i. Il est toutefois

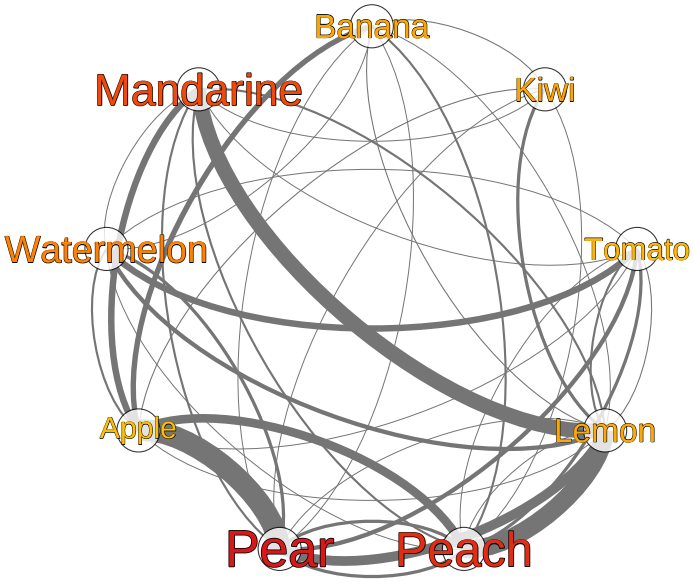


FIGURE 4.2 : Cas d'étude avec le « thème des fruits ». La direction des arcs est donnée par le sens de rotation des aiguilles d'une montre. L'intensité de la couleur des termes reflète la valeur du critère défini à l'équation 4.2.

légitime de penser que l'apparition de certains termes du lexique est plus significative du thème des documents. Par conséquent, nous souhaitons intégrer les poids des termes t_j dans le calcul du poids du terme t_i. Partant de l'équation 4.2, nous remplaçons simplement le numérateur en y intégrant les poids de t_j et en les normalisant par leur degré sortant, ce qui aura pour effet de distribuer équitablement le poids de t_j à tous ses successeurs. Formellement, le poids w_i d'un terme t_i est enfin défini comme :

$$w_i = \sum_{t_j \in \mathcal{L}_T^\dagger} \frac{n_{t_j,C_i} \cdot w_j}{\sum_{k \in \{1,\dots,|\mathcal{L}_T|\}} n_{t_j,C_k}} \tag{4.3}$$

L'équation 4.3 est en fait équivalente à l'algorithme de marche aléatoire PageRank (Page et coll., 1999). Ce dernier peut être résolu par la méthode de la puissance itérée et converge vers une solution unique sous réserve des conditions suivantes (Langville et Meyer, 2005; Farahat et coll., 2006) : *(i)* la matrice d'adjacence du graphe doit être stochastique [5], ce qui garantit l'existence d'une ou plusieurs solutions ; *(ii)* le graphe doit être fortement connecté, ce qui garantit l'unicité de la solution.

Pour obtenir une matrice stochastique, il est nécessaire de modifier les sommets sans arc sortant (ou *dangling nodes*) dont le degré sortant vaut 0.

5. C'est-à-dire que toutes ses lignes doivent sommer à 1.

Cette modification semble logique, car il n'est pas souhaitable qu'un terme n'apparaissant dans aucun des corpus agrège un poids important. Nous modifions notre graphe et ajoutons un lien de chaque sommet sans arc sortant vers un sommet virtuel et un lien de ce sommet virtuel vers tous les sommets du graphe. Le poids des sommets sans arc sortant est ainsi redistribué uniformément à tous les sommets du graphe.

Pour assurer l'unicité de la solution, nous normalisons tout d'abord les poids des arcs du graphe pour obtenir une chaîne de Markov. Cette normalisation garantit par ailleurs que l'équation 4.3 fournit en sortie une distribution de probabilité où chaque sommet est pondéré en fonction de la probabilité d'atterrir sur ce sommet en suivant aléatoirement les arcs du graphe pendant un grand nombre de pas. Puis, nous appliquons la solution de Page et coll. (1999) et ajoutons une probabilité de téléportation uniforme à chaque itération de l'algorithme ce qui garantit la forte connexité du graphe et la convergence vers une solution unique. Nous obtenons ainsi l'équation finale suivante :

$$w_{i,n+1} = \frac{(1-\alpha)}{N} + \alpha \cdot \sum_{t_j \in \mathcal{L}_T^\dagger} \frac{n_{t_j,C_i} \cdot w_j}{\sum_{k \in \{1,\dots,|\mathcal{L}_T|\}} n_{t_j,C_k}} \tag{4.4}$$

où N est le nombre de sommets du graphe (c'est-à-dire le nombre de termes du lexique) et α est un facteur d'amortissement traditionnellement fixé à 0,85 (Page et coll., 1999). Nous utilisons également cette valeur de α pour nos expériences.

La Figure 4.3 présente les résultats obtenus sur l'exemple précédent (Figure 4.2) avec l'équation 4.4. Nous constatons que les résultats obtenus sont bien différents de ceux obtenus avec le degré entrant : l'importance du terme « Pear » a été revue à la baisse, car le terme « Apple » est faiblement pondéré. Le terme « Mandarine » apparaît peu dans les différents corpus (degré sortant faible), mais contient de nombreux termes pertinents dans son propre corpus (degré entrant élevé). C'est donc un terme relativement rare et pertinent. L'apparition d'un tel terme dans le corpus du terme « Lemon » confère à ce dernier un poids important, ce qui n'était pas le cas du critère précédent fondé sur le degré entrant. Cet exemple illustre l'intérêt de l'algorithme PageRank : une mesure *globale* de l'importance d'un sommet dans un graphe.

L'algorithme 1 récapitule l'intégralité du calcul de pertinence thématique des termes d'un lexique.

Notons enfin que pour améliorer la correspondance entre les lexiques thématiques et les documents issus du Web, nous appliquons une série

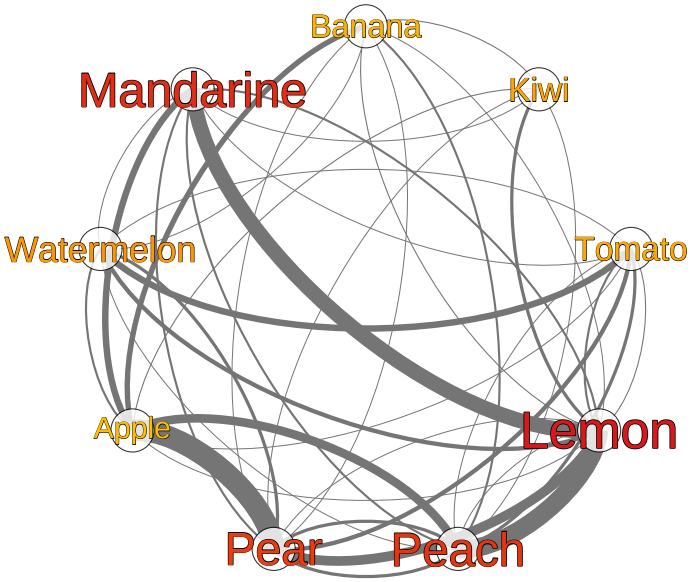


FIGURE 4.3 : Cas d'étude avec le « thème des fruits ». L'intensité de la couleur des termes reflète la valeur du critère défini à l'équation 4.4.

de normalisations supplémentaires : conversion des termes en minuscules, racinisation (Porter, 1980) et normalisation des caractères Unicode (suppression des diacritiques).

4.3 Évaluations

Avant d'évaluer l'apport de la cohésion thématique pour la sélection de requêtes, nous nous intéressons à la sélection des paramètres optimaux du critère. Pour ce faire, nous évaluons empiriquement l'influence du nombre de documents (M), de la taille du lexique (N) ou encore l'utilité des *snippets* sur les poids produits par le critère. Nous réalisons ces expériences sur un ensemble de lexiques de référence sur trois thèmes distincts : l'Astronomie, le Médical et les Statistiques.

Une fois ces paramètres sélectionnés, nous évaluons l'apport du critère pour la sélection de requêtes au préalable d'une phase de constitution de corpus. Nous utilisons alors les méthodes et corpus décrits précédemment (§ 3.3, p. 40), en plus de mesures de corrélation entre les poids produits et la précision moyenne des requêtes. Enfin, nous étudions l'influence du choix du moteur de recherche sur les poids produits.

Algorithme 1 Mesure de pertinence d'un terme pour un thème

Entrées : $\mathcal{L}_T$: termes $t_i, i \in [1, N]$
M : nombre de documents téléchargés par requête
α : facteur d'amortissement

// Téléchargement du corpus
Pour tout terme $t_i \in \mathcal{L}_T$ **faire**
  Soumettre t_i à un moteur de recherche
  Télécharger M documents/*snippets* comme corpus C_i
Fin Pour

// Initialisation
Pour tout terme $t_i \in \mathcal{L}_T$ **faire**
  $w_{i,1} = 1/N$
Fin Pour

// Procédure itérative de calcul des poids
$n = 1$
Tant que (non-convergence) **faire**
  Pour tout terme $t_i \in \mathcal{L}_T$ **faire**
    $$w_{i,n+1} = \frac{(1-\alpha)}{N} + \alpha \cdot \sum_{t_j \in \mathcal{L}_T^\dagger} \frac{n_{t_j,C_i} \cdot w_j}{\sum_{k \in \{1,\ldots,|\mathcal{L}_T|\}} n_{t_j,C_k}}$$
  Fin Pour
  Normalisation des poids : $\sum_i w_{i,n+1} = 1$
  $n = n + 1$
Fin Tant que
Retourner w_n

4.3.1 *Évaluation de l'influence des paramètres du critère*

DONNÉES

Pour cette expérience, nous nous appuyons sur trois thésaurus en anglais :

- Astronomie (*The Astronomy Thesaurus*[6]) ;
- Médical (*Medical Subject Headings* ou MeSH[7]) ;
- Statistiques (*International Statistical Institute*[8]).

6. http://msowww.anu.edu.au/library/thesaurus/
7. Nous travaillons sur un extrait du MeSH (http://www.nlm.nih.gov/mesh/) sélectionné à partir de la version 2008aa de l'UMLS (http://www.nlm.nih.gov/research/umls/) en ne conservant que les termes vedettes (*Main Heading*).
8. http://isi.cbs.nl/glossary/

Astronomie	Statistiques	Médical
afterglow	Birnbaum's inequality	wandering spleen
celestial coordinates	geometric mean	dimethoxyphenylethylamine
asteroids	K-test	wolman disease
dwarf stars	invariant	antimalarials
bow shocks	cross spectrum	optical illusions
x rays	fertility rate	privacy
films	area sampling	hyphema
Einstein shift	dendrogram	false negative reactions
red dwarf stars	antiseries	mephenytoin
auroral jets	gamma distribution	nobel prize

Tableau 4.1 : Extrait des lexiques thématiques Astronomie, Statistiques et Médical

Un extrait de ces lexiques est présenté au Tableau 4.1.

Une série de traitements a été appliquée à chaque lexique dans le but d'en améliorer la qualité ou l'utilisation pour notre évaluation. Ainsi, nous avons supprimé les termes apparaissant entre crochets ou parenthèses dans les lexiques Astronomie et Statistiques. Le lexique Médical présentant des termes trop ambigus pour être nettoyé automatiquement (comme, par exemple, *3-pyridinecarboxylic acid, 1,4-dihydro-2,6-dimethyl-5-nitro-4-(2-(trifluoromethyl)phenyl)-, methyl ester*), nous avons simplement supprimé les termes contenant une parenthèse ou une virgule. Le lexique Médical comprenant plus de 19 000 termes, nous avons choisi de ne travailler que sur un échantillon de ce dernier. Nous avons donc tiré aléatoirement deux séries de 2 000 termes que nous nommons lexiques Médical-1 et Médical-2.

Nous obtenons finalement les lexiques suivants :
– Astronomie (2 940 termes) ;
– Statistiques (2 752 termes) ;
– Médical-1 (2 000 termes) ;
– Médical-2 (2 000 termes).

MOTEUR DE RECHERCHE

Cette expérience sur les thèmes de l'Astronomie, du Médical et des Statistiques étant indépendant de nos autres travaux sur l'OpenDirectory, nous nous sommes appuyé sur le moteur de recherche Blekko [9] pour constituer les corpus nécessaires à l'application de notre méthode. Blekko est un

9. http://www.blekko.com

moteur de recherche Web anglophone qui possède un index de 4 milliards de pages Web [10]. Leur API de recherche permet de télécharger au maximum 500 résultats pour chaque requête. Ces résultats incluent les URL des pages Web ainsi que les *snippets* associés. Ces *snippets* sont composés en moyenne de 36 tokens (243 caractères) issus du corps des pages Web et contenant le ou les termes de la requête.

Pour 14% des termes du lexique Statistiques, aucun document n'a été retourné. Les lexiques Astronomie, Médical-1 et Médical-2 ont quant à eux obtenu un taux d'échec de moins de 3 %.

NOMBRE DE DOCUMENTS ET NOMBRE DE SNIPPETS

Nous souhaitons évaluer l'influence du nombre de documents ou de *snippets* téléchargés sur le critère proposé. Pour ce faire, nous appliquons la méthode décrite à l'Algorithme 1 en faisant varier le paramètre M :

- Pour les documents, nous fixons M à 10, 20, 50, et 100 documents téléchargés.
- Pour les *snippets*, nous faisons varier M entre 50 et 500 *snippets* téléchargés avec un pas de 50.

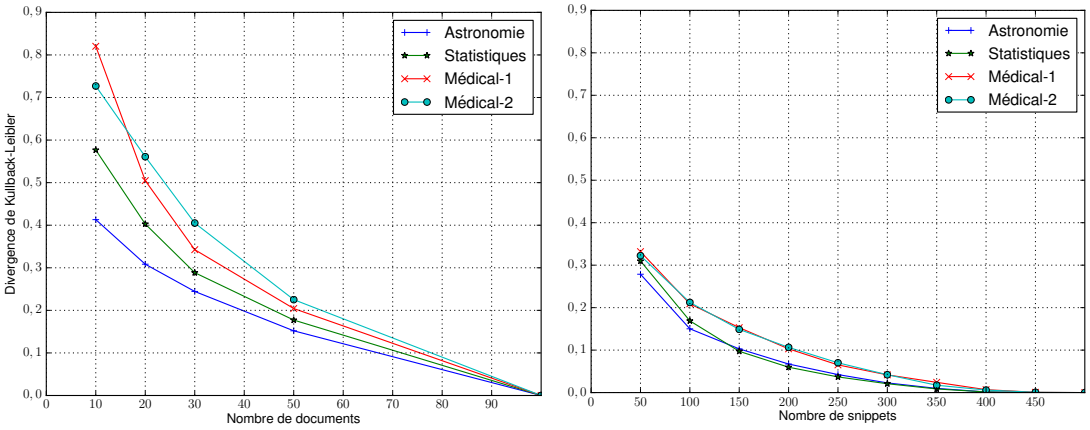
Nous comparons les poids des termes obtenus pour un nombre de documents (resp. *snippets*) maximal et ceux obtenus avec un nombre de documents (resp. *snippets*) inférieur. Comme les poids fournis par notre critère représentent une distribution de probabilités, nous utilisons la divergence de Kullback-Leibler (Kullback et Leibler, 1951) pour mesurer leur différence. La divergence de Kullback-Leibler est définie pour deux distributions de probabilité discrètes p et q comme :

$$D(p,q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}$$

Elle produit un score dans $\mathbb{R}^+$ et vaut zéro pour deux distributions p et q identiques.

La Figure 4.4 présente la divergence de Kullback-Leibler obtenue avec les pages Web (4.4a) et les *snippets* (4.4b). Alors que les résultats obtenus avec moins de 30 documents offrent une divergence supérieure à $0,4$, les *snippets* montrent une plus grande stabilité avec une divergence inférieure dès 50 *snippets* téléchargés.

10. http://blog.blekko.com/2012/07/03/blekko-upgrades-seo-data/



(a) Documents (b) *Snippets*

FIGURE 4.4 : Divergence de Kullback-Leibler entre les résultats obtenus avec le nombre maximum de documents et un nombre de documents inférieur.

DOCUMENTS ET SNIPPETS

Nous évaluons à présent la différence de résultats obtenue lorsque l'on utilise des documents ou des *snippets* comme source d'information. Pour ce faire, nous utilisons à nouveau la divergence de Kullback-Leibler pour comparer les poids obtenus en appliquant le critère proposé avec chacune des méthodes. La Figure 4.5 présente ces résultats pour le lexique « Astronomie ». Les résultats obtenus pour les autres lexiques sont similaires et vérifient également les conclusions suivantes. Nous constatons tout d'abord que la divergence décroît rapidement avec l'augmentation du nombre de documents. Logiquement, la divergence minimale est obtenue pour le nombre maximal de documents (100) et de *snippets* (500). Il est intéressant de noter que la divergence est relativement stable dès que plus de 100 *snippets* sont utilisés. Elle est inférieure à $0,2$ pour 100 *snippets* et décroît à $0,16$ pour 200 *snippets*. Par conséquent, peu de *snippets* suffisent à approcher le résultat obtenu avec 100 documents. Étant donné que le téléchargement et le nettoyage de pages Web sont des traitements coûteux, il semble donc opportun de se limiter au téléchargement des *snippets*. Par conséquent, nous utilisons cette source d'information pour la suite de nos expériences et fixons le nombre de *snippets* à 200.

NOMBRE DE TERMES DANS LE LEXIQUE

Le poids d'un terme t_i est défini en fonction des cooccurrences de ce terme avec les autres termes du lexique (équation 4.4). Il semble donc légitime de s'interroger sur la stabilité des poids des termes en fonction de la taille du lexique. La somme des poids étant égale à 1, le poids absolu

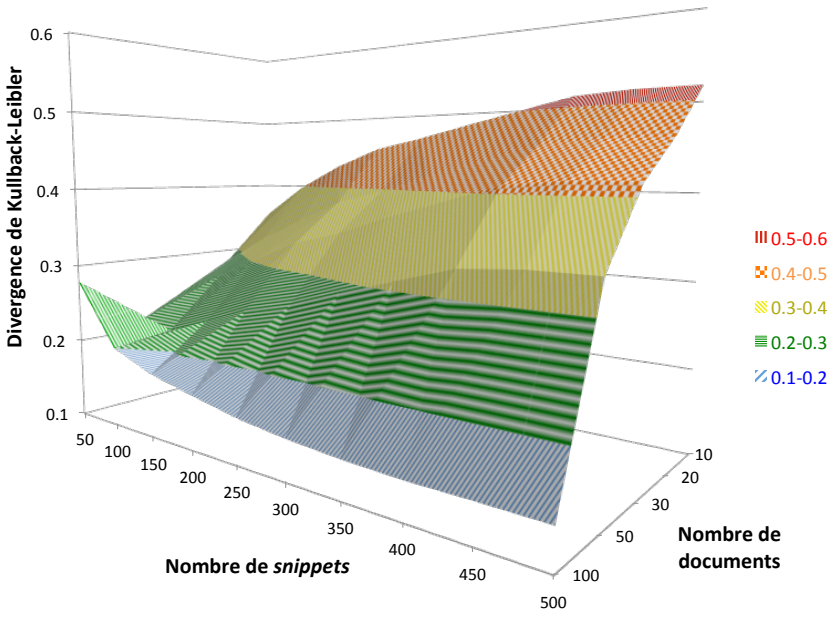


FIGURE 4.5 : Divergence de Kullback-Leibler entre les résultats obtenus avec les *snippets* ou les documents pour le lexique « Astronomie ».

de chaque terme est donc lié à la taille du graphe : il va diminuer avec l'augmentation du nombre total de termes. La question est donc de savoir ce qu'il en est du poids relatif, c'est-à-dire du rang des termes, en fonction de la taille du lexique.

Pour cette évaluation, nous avons appliqué la procédure suivante 5 fois pour chacun des lexiques, puis nous avons calculé une moyenne des résultats :

- Nous sélectionnons aléatoirement 20 termes ;
- Itérativement nous augmentons le nombre de termes avec un pas de 20 jusqu'à obtenir un lexique de 1000 termes ;
- À chaque étape, nous évaluons l'évolution des rangs des N termes du lexique entre eux.

Nous avons besoin d'une mesure de distance pour évaluer l'évolution des rangs des termes entre lexiques successifs (20-40, 40-60, ...). Cette mesure doit être indépendante du nombre de termes dans les lexiques. Nous utilisons la distance de Spearman normalisée (*normalized Spearman's Footrule Distance* (Diaconis, 1988; Dwork et coll., 2001)), définie comme suit : étant donné deux listes d'éléments pondérés A et B, nous posons α

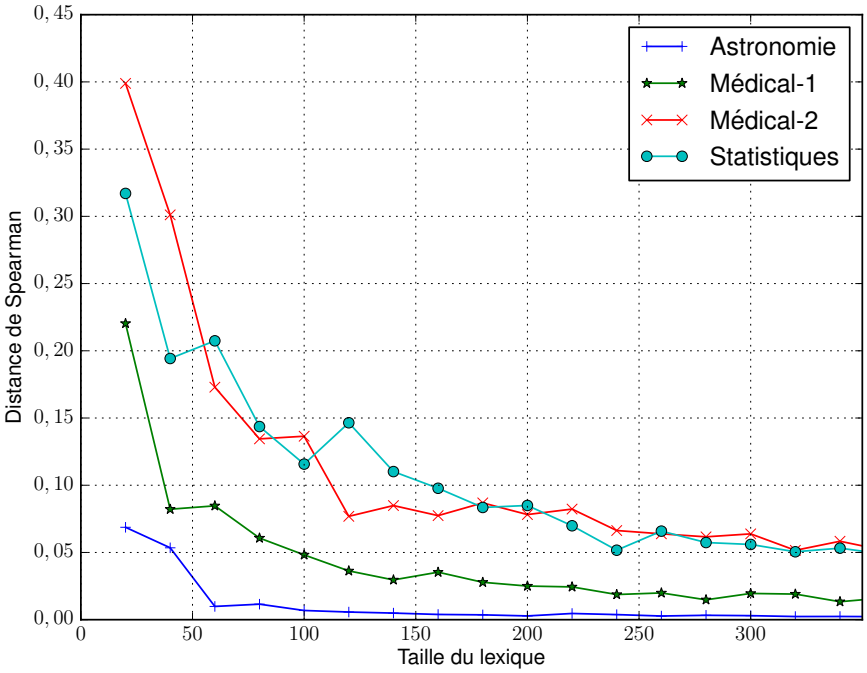


FIGURE 4.6 : Distance de Spearman normalisée entre les poids obtenus pour des lexiques de différentes tailles.

et β comme les rangs des termes de chaque lexique. Alors, la distance de Spearman normalisée entre 0 et 1 vaut :

$$D(\alpha, \beta) = \frac{\sum_{t_i \in \mathcal{L}_T} |\alpha(t_i) - \beta(t_i)|}{|\mathcal{L}_T|^2/2}$$

où $\mathcal{L}_T$ est notre lexique thématique. En d'autres termes, cette mesure est simplement la somme normalisée des différences entre les rangs des éléments dans les deux listes.

La Figure 4.6 présente les résultats de cette expérience. Nous observons des variations importantes avec peu de termes, qui s'amenuisent ensuite sensiblement à partir de 100 termes. Nous en concluons que les poids obtenus par notre critère varient pour des lexiques de petite taille (inférieurs à 100 termes) mais deviennent stables pour des lexiques de taille supérieure. En outre, nous constatons qu'une taille de lexique supérieure à 250 termes ne permet pas d'améliorer la qualité des poids des termes.

4.3.2 *Évaluation du critère pour la sélection de requêtes*

Après nous être intéressés à l'influence des paramètres du critère sur des lexiques non bruités, nous souhaitons évaluer l'apport du critère pour la sélection de requêtes. Ainsi, nous appliquons notre mesure pour réordonner les termes amorces précédemment extraits des descriptions de catégories de l'OpenDirectory (déjà utilisés § 3.2, p. 40). En accord avec nos dernières conclusions, nous sélectionnons les 200 termes ayant le meilleur *tfidf* pour les repondérer ensuite à l'aide de notre critère. Pour chaque requête, nous téléchargeons les 200 premiers *snippets* renvoyés par le moteur de recherche. Nous nous appuyons sur deux moteurs de recherche : le moteur de recherche Web Blekko et le moteur de recherche que nous utilisons pour la recherche orientée Lucene. Dans les deux cas, nous utilisons des *snippets* de longueur similaires (250 caractères). Nous faisons l'hypothèse qu'utiliser le même moteur de recherche pour la prédiction de performances et la collecte de documents spécialisés devrait amener de meilleures performances. Malgré cela, il nous semble intéressant d'évaluer le biais dû à l'utilisation d'un autre moteur de recherche, qui pourrait être installé localement et offrir ainsi un temps de traitement plus faible.

Tout d'abord, nous évaluons la validité de notre critère de cohésion thématique pour la prédiction de performances d'un unique terme. Puis, nous poursuivons cette évaluation en appliquant notre critère pour la prédiction de performances d'une requête. Nous envisageons deux méthodes : *(i)* sélectionner les 10 meilleurs termes amorces à l'aide de notre critère et réaliser toutes les combinaisons de ces termes ; *(ii)* combiner les scores prédits pour chaque terme afin de fournir une prédiction de performances pour chaque requête.

IMPACT DU RÉORDONNANCEMENT

Nous étudions en premier lieu le recouvrement entre les termes amorces sélectionnés par l'heuristique du *tfidf* et ceux choisis par le critère proposé. Notre objectif est ici de mesurer l'impact du réordonnancement sur le choix des amorces. Dans le cadre du moteur de recherche Blekko, $2,1$ termes en moyenne apparaissent simultanément parmi les 10 meilleurs termes fournis par le *tfidf* et notre mesure. Au contraire, avec le moteur de recherche Lucene, seuls $1,3$ termes apparaissent conjointement avec les deux mesures. Les termes amorces obtenus avec chaque mesure sont donc très différents, ce qui nous permet d'espérer un impact important sur les performances de collecte.

Mesure	Spearman ρ	Significativité (P < 0,05)
tfidf	0,200 (±0,164)	32%
Cohésion thématique (Blekko)	0,434 (±0,208)	74%
Cohésion thématique (Lucene)	0,555 (±0,186)	88%

Tableau 4.2 : Coefficient de corrélation de Spearman entre la prédiction (cohésion thématique) et le résultat de la requête (précision moyenne) pour les catégories du second niveau de l'OpenDirectory.

CORRÉLATION AVEC LA PRÉCISION MOYENNE

Nous souhaitons à présent valider notre mesure de prédiction de performances pour chaque terme. Pour ce faire, nous soumettons chaque terme en requête et stockons la précision moyenne des résultats obtenus pour cette dernière. La précision moyenne permet d'évaluer non seulement la pertinence des documents renvoyés mais également la qualité de l'ordre dans lequel ils sont renvoyés. Elle est calculée comme la moyenne des valeurs de précision obtenues après chaque document pertinent trouvé. Ou, plus formellement,

$$\text{PrécisionMoyenne} = \frac{\sum_{k=1}^{n}(\text{Précision}(R_k) \times \text{Pertinent}(k))}{\text{nombre de documents pertinents}}$$

où R_k est l'ensemble des documents renvoyés ordonnés jusqu'au rang k et $\text{Pertinent}(k)$ est une fonction indicatrice de la pertinence du document au rang k.

Nous mesurons ensuite la corrélation entre la précision moyenne et les poids fournis par le *tfidf* ou la cohésion thématique. Comme ce sont les rangs des requêtes qui nous intéressent ici et non les valeurs prédites, nous optons pour le coefficient de corrélation de Spearman. Ce coefficient de corrélation est défini comme suit : étant donné deux listes d'éléments pondérés A et B, nous posons α et β comme les rangs des termes pour chacune des listes. Alors, le coefficient de corrélation de Spearman ρ vaut :

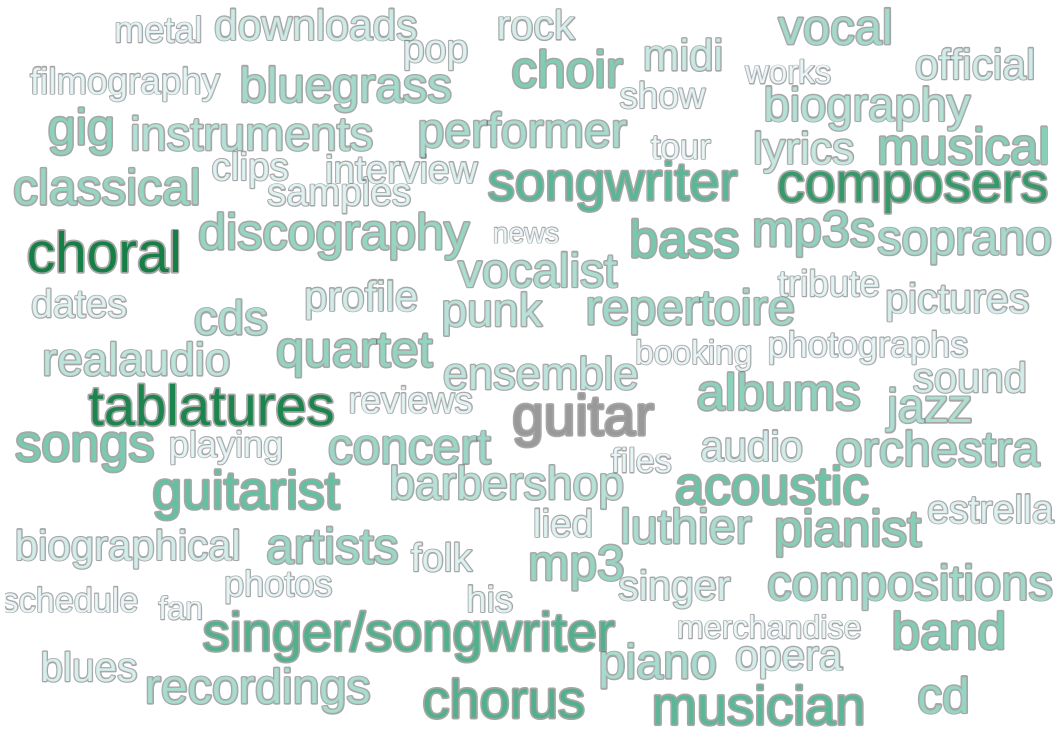
$$\rho(\alpha, \beta) = 1 - \frac{6\sum_{t_i \in \mathcal{L}_T}(\alpha(t_i) - \beta(t_i))^2}{|\mathcal{L}_T|(|\mathcal{L}_T|^2 - 1)}$$

Ce coefficient prend une valeur entre -1 et 1 indiquant le degré de dépendance entre les deux ordres. Plus le coefficient est proche des valeurs extrêmes et plus la corrélation est forte.

Nous présentons les résultats au Tableau 4.2. Étant donné que nous travaillons sur l'ensemble des catégories du second niveau de l'OpenDirectory, nous calculons une macro moyenne des coefficients de corrélation.

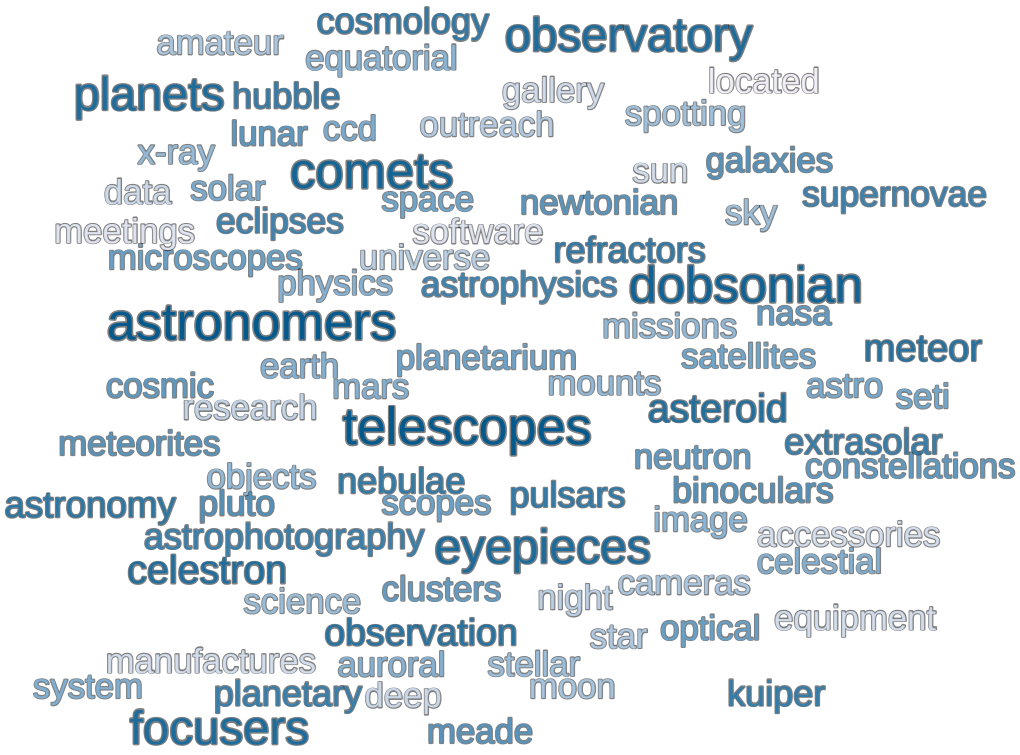
Nous évaluons également le pourcentage de catégories pour lesquelles la corrélation est statistiquement significative (p-valeur inférieure à $0,05$). Nous constatons que la mesure du *tfidf* offre une corrélation faible avec la précision moyenne et n'est, en outre, significative que pour 32% des catégories. Au contraire, la mesure de cohésion thématique offre une corrélation moyenne à forte en fonction du moteur de recherche de recherche utilisé. En utilisant un moteur de recherche Web extérieur (ici, Blekko) la corrélation est de $0,43$ alors qu'en utilisant notre index local de l'OpenDirectory (Lucene), la corrélation est logiquement plus forte ($0,53$). Notons enfin que les résultats obtenus avec le moteur de recherche Blekko et ceux obtenus avec Lucene sont fortement corrélés (0.576 (± 0.148)) ce qui montre une certaine indépendance vis-à-vis du moteur de recherche tant que le nombre de documents est suffisant.

Pour illustrer la qualité des termes extraits par notre critère, nous présentons quatre exemples de graphes pondérés par notre critère à la Figure 4.7. Chaque graphe a été construit à partir des lexiques extraits des descriptions de catégories de l'OpenDirectory. Pour plus de clarté, les arcs et les sommets des graphes ont été cachés, ce qui revient à une représentation de type « nuage de mots ».



(a) Arts/Music

FIGURE 4.7 : Illustration de l'application de notre critère aux termes issus des descriptions de l'OpenDirectory. La taille des termes et l'intensité des couleurs symbolisent le poids obtenu par la cohésion thématique.



(b) Science/Astronomy

(c) Home/Cooking

(d) Society/Crime

UTILISATION POUR LA SÉLECTION DE TERMES AMORCES

Nous évaluons à présent l'apport de notre critère pour la sélection de termes amorces. Nous ordonnons l'ensemble des 200 termes candidats par notre critère et sélectionnons les 10 meilleurs termes comme amorces. Nous utilisons à présent le même moteur de recherche que celui employé pour la recherche orientée, c'est-à-dire Lucene. Nous formulons ensuite des requêtes composées de 1, 2 ou 3 termes parmi les 10 sélectionnés. Nous évaluons comme précédemment (Tableau 3.6, § 3.4, p. 44) la précision et le rappel des documents téléchargés.

Les résultats sont présentés au Tableau 4.3. Nous avions précédemment constaté que notre critère est positivement corrélé avec la mesure de précision moyenne pour un unique terme. Cette nouvelle expérience valide cette hypothèse en montrant une hausse de la précision et du rappel pour les requêtes composées d'un unique terme. Toutefois, nous pouvons constater que le gain en précision s'amenuise avec l'augmentation de la taille des requêtes. De plus, le rappel, déjà faible, diminue avec l'utilisation de notre critère pour des requêtes composées de deux termes et plus. Nous pouvons conclure que la sélection de termes amorces par notre critère permet d'améliorer la précision de la collecte. Cependant, il semble que les termes choisis mènent à un nombre de documents pertinents plus

Nb termes	Nb requêtes	Précision	Rappel
1	10	0,366 (+0,150)	0,038 (+0,013)
2	45	0,395 (+0,066)	0,052 (-0,032)
3	120	0,420 (+0,067)	0,036 (-0,065)

Tableau 4.3 : Macro moyenne de la précision et du rappel pour des requêtes de tailles différentes construites à partir de termes amorces sélectionnés par notre critère.

faibles. Il sera donc nécessaire de formuler un plus grand nombre de requêtes pour obtenir un corpus thématique de même taille.

UTILISATION POUR LA SÉLECTION DE REQUÊTES

Nous venons de constater que la sélection de bons termes amorces procure un gain en précision sur les requêtes composées de plusieurs termes. Néanmoins, nous voudrions prédire directement la performance de combinaisons de termes pour pouvoir sélectionner de bonnes requêtes parmi toutes les combinaisons possibles. Pour aboutir à ce résultat sans soumettre pour autant toutes les requêtes correspondant aux combinaisons de termes, nous proposons de « combiner » les scores obtenus pour chaque terme. Nous inspirant simplement de la théorie des probabilités qui stipule que pour deux évènements A et B indépendants, $P(A \cap B) = P(A) \times P(B)$, nous proposons simplement de multiplier les scores fournis par notre critère pour chacun des termes afin de fournir un score à chaque requête. Formellement, pour une requête $Q = \{q_i\}_{i=1}^{N}$, nous calculons $CT(Q) = \prod_{i=1}^{N} CT(q_i)$ où CT est le score fourni par notre critère. Pour évaluer cette approche, nous appliquons simplement cette mesure pour ordonner l'ensemble des requêtes et sélectionnons le même nombre de requêtes que précédemment. Nous mesurons ensuite une nouvelle fois la précision et le rappel de la collecte. Nous présentons les résultats au Tableau 4.4 et observons à nouveau un gain supplémentaire en précision et une légère baisse en rappel (présentés entre parenthèses) par rapport aux résultats initiaux (Tableau 3.6).

4.4 Bilan

Dans ce chapitre, nous nous sommes intéressés à la définition d'une mesure de prédiction des performances des termes et combinaisons de termes pour la recherche orientée. Nous avons tout d'abord défini une mesure de cohésion thématique qui ne nécessite en entrée qu'un ensemble de termes

Nb termes	Nb requêtes	Précision	Rappel
1	10	0,366 (+0,150)	0,038 (+0,013)
2	45	0,408 (+0,079)	0,049 (-0,035)
3	120	0,447 (+0,094)	0,033 (-0,068)

Tableau 4.4 : Macro moyenne de la précision et du rappel pour des requêtes de tailles différentes sélectionnées par notre critère.

du domaine. Cette mesure est calculée en deux temps : un ensemble de documents contenant chaque terme est collecté à partir d'un moteur de recherche. Puis, nous modélisons les cooccurrences de ces termes sous la forme d'un graphe orienté et appliquons un algorithme de marche aléatoire afin d'assigner à chaque terme une mesure globale de son importance pour le thème.

Nous avons évalué l'influence de différents paramètres tels que le nombre de documents, l'utilisation de *snippets* ou le nombre de termes, sur notre mesure. Pour ce faire, nous nous sommes appuyés sur trois lexiques thématiques de référence traitant d'Astronomie, du domaine Médical ou des Statistiques. Nous avons constaté que peu de *snippets* permettaient d'approximer correctement le résultat obtenu avec de nombreux documents. De plus, ces derniers sont beaucoup plus rapides à collecter, car une simple requête au moteur de recherche suffit.

Mentionnons en passant que nous avons également appliqué notre mesure sur ces mêmes lexiques traduits automatiquement *via* l'outil Google Translate [11], pour aider des utilisateurs à créer des lexiques thématiques bilingues (de Groc et coll., 2012). Nous avions alors évalué la supériorité de la version finale de notre critère de cohésion thématique (équation 4.4) sur sa version initiale (équation 4.2). Ce sont également ces expériences qui nous ont incité à intégrer le nombre d'occurrences des termes dans les corpus à l'équation 4.2 plutôt que le nombre de documents.

Nous avons ensuite appliqué notre critère pour la sélection de termes amorces et de requêtes. Pour ces expériences, nous nous appuyons comme précédemment sur le moteur de recherche Lucene et l'ensemble des 340 catégories du second niveau de l'OpenDirectory. Nous avons observé une corrélation forte entre notre critère et la précision moyenne des requêtes composées d'un unique terme. Ce résultat montre que notre critère capture correctement la pertinence thématique des termes. Nous avons en-

11. http://translate.google.com

suite proposé d'utiliser notre critère pour sélectionner les termes amorces et les requêtes. Dans les deux cas, nous avons observé un gain en précision, accompagné d'une baisse du rappel. Ces résultats sont encourageants, car nous souhaitons mettre l'accent sur la précision de la collecte. Néanmoins, un rappel plus faible signifie qu'il sera nécessaire de définir un plus grand nombre de termes amorces et de requêtes pour obtenir une couverture équivalente du domaine.

Une piste de recherche qui nous semble pertinente serait d'intégrer une notion de dispersion des requêtes dans notre critère. Notre objectif au travers de cette approche serait de favoriser les requêtes renvoyant un grand nombre de documents distincts et ainsi d'augmenter le rappel. Cet objectif pourrait également être atteint en post-traitement, en combinant notre critère à une seconde mesure de dispersion.

5 GRAWLTCQ : GRAPHES HÉTÉROGÈNES ET MARCHES ALÉATOIRES POUR LA COLLECTE SPÉCIALISÉE

Dans ce chapitre, nous nous intéressons à l'application de traitements en aval du moteur de recherche afin d'améliorer la qualité des corpus produits. En effet, nous avons constaté des performances mitigées au Tableau 3.6 (p. 44) que nous améliorons légèrement en sélectionnant de meilleurs termes amorces (Tableau 4.4, p. 70). Par conséquent, nous nous intéressons au filtrage des corpus produits afin d'en améliorer encore la qualité.

Une méthode particulièrement efficace pour le filtrage des corpus produits serait d'appliquer un catégoriseur thématique que l'on aurait préalablement entraîné sur un petit corpus annoté, à l'instar de Ghani et coll. (2005). La catégorisation automatique de texte est un domaine de recherche très étudié depuis une vingtaine d'années, et offre de bonnes performances dans un cadre binaire (Sebastiani et Ricerche, 2002). Cependant, dans notre cas d'utilisation, nous ne disposons pas nécessairement d'un corpus thématique existant pour entraîner un catégoriseur, mais simplement d'un petit ensemble de termes du domaine. Par conséquent, nous allons nous intéresser à des méthodes de filtrage non supervisées.

Notre objectif est de partitionner le corpus produit et de ne conserver que le sous-ensemble traitant du thème voulu. Plutôt qu'effectuer un partitionnement strict des documents, nous proposons d'attribuer un poids à chaque document en fonction de son attachement au thème, ou plus précisément, aux termes amorces qui représentent notre thème. Pour ce faire, nous proposons une méthode, dénommée GrawlTCQ[1] (de Groc et coll., 2011b), fondée sur un graphe *hétérogène* incluant termes amorces, requêtes, documents et les termes extraits de ces documents. Nous appliquons ensuite un algorithme de marche aléatoire biaisé (*random walk with restart*) envers les termes amorces, pour pondérer simultanément les termes, les requêtes et les documents. Le poids fourni à chaque sommet représente alors l'importance *globale* du sommet dans le graphe. Nous ordonnons enfin les sommets de même type (documents, termes) pour obtenir l'importance relative d'une entité. L'ordre produit pour les documents peut

1. Pour *Graph RAndom WaLk for Terminology, Corpora and Queries.*

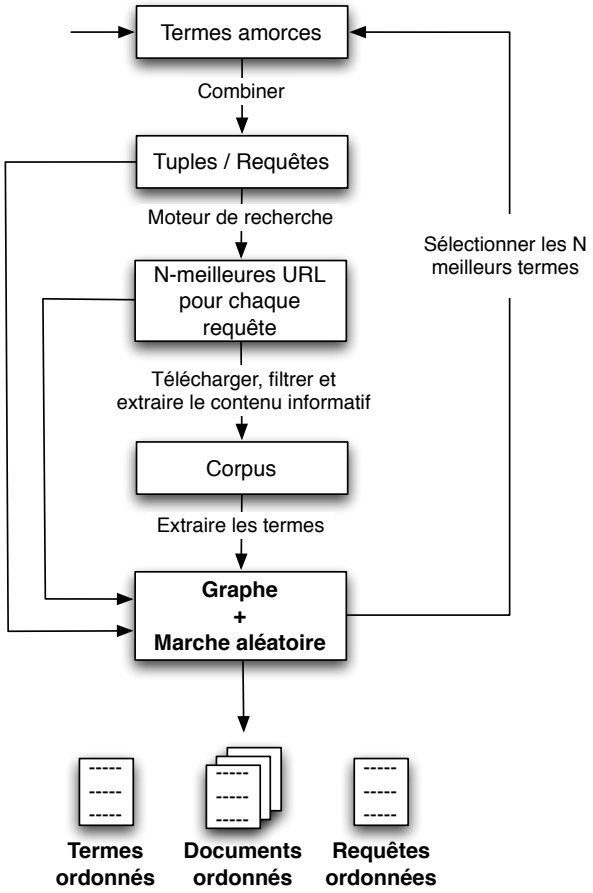


FIGURE 5.1 : Procédure GrawlTCQ, graphe hétérogène et marche aléatoire pour la sélection de documents et de termes.

finalement être utilisé pour sélectionner les documents les plus pertinents, alors que l'ordre produit pour les termes peut permettre de sélectionner de nouveaux termes amorces pertinents en vue de soumettre de nouvelles requêtes. La procédure proposée est schématisée à la Figure 5.1.

5.1 Travaux liés

Depuis quelques années, les algorithmes d'ordonnancement fondés sur une modélisation graphique ont reçu un intérêt de plus en plus important des communautés de recherche d'information et de traitement automatique des langues. L'algorithme du PageRank a en particulier été appliqué à des domaines variés : pour mesurer l'importance de pages internet (Page et coll., 1999), extraire des termes ou des phrases saillantes (Mihalcea et Tarau, 2004), catégoriser des textes (Hassan et coll., 2006), pour la désambiguïsation sémantique (Agirre et Soroa, 2009) ou l'expansion d'ensembles d'entités nommées (Wang et Cohen, 2007), comme pondération dans un

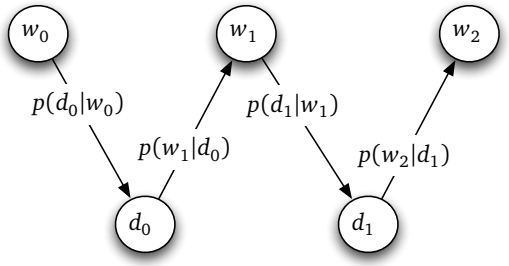


FIGURE 5.2 : Exemple de graphe utilisé par Lafferty et Zhai (2001) pour l'expansion de requêtes ou de corpus.

modèle de recherche d'information (Blanco et Lioma, 2007) ou encore pour l'expansion de requêtes (Lafferty et Zhai, 2001; de Groc et Tannier, 2012).

Nous nous intéressons plus précisément ici aux méthodes se fondant sur des graphes dits *hétérogènes*. Lafferty et Zhai (2001) ont utilisé une chaîne de Markov composée de termes et documents pour l'expansion de requêtes et de corpus. Leur algorithme de marche aléatoire alterne la sélection d'un document et d'un terme en fonction des probabilités issues de modèles de langue (Figure 5.2). Un graphe similaire est utilisé par Dhillon (2001) pour la tâche de *coclustering*. Ici, l'algorithme de segmentation spectrale est équivalent à un algorithme de marche aléatoire (Maila et Shi, 2001) et permet de regrouper simultanément les documents et termes entre eux.

Wang et Cohen (2007) ont proposé d'utiliser un graphe hétérogène et un algorithme de marche aléatoire pour étendre un petit ensemble d'entités nommées. Leur graphe inclut les requêtes, documents, entités nommées et le contexte HTML d'apparition de ces entités dans les documents (*wrappers*). Minkov et coll. (2006) utilisent un graphe hétérogène pour modéliser des informations relatives aux e-mails (expéditeur, destinataire, adresses, fichiers, corps du message). Enfin, Nie et coll. (2005) décrivent une méthode pour la recherche bibliographique dans Microsoft Libra[2]. Ils introduisent différentes informations bibliographiques (conférences/revues, articles, auteurs, etc.) comme sommets de leur graphe et pondèrent les arcs du graphe automatiquement : les poids sont déterminés par un algorithme d'optimisation ayant pour objectif de minimiser la distance entre l'ordre produit par l'algorithme de marche aléatoire et un ensemble de données partiellement annotées (une liste de quelques conférences du domaine ordonnées par importance, par exemple).

2. Maintenant nommé *Microsoft Academic Search*.

5.2 Modèle de graphe

Nous modélisons la procédure de recherche orientée sous la forme d'un graphe. Les sommets du graphe sont composés des trois types d'*entités* : les *termes*, les *requêtes* et les *documents*. Ces derniers sont liés entre eux par des arcs orientés typés qui dénotent la relation entre ces entités : $\texttt{in\_query}(\texttt{t}, \texttt{q})$ si le terme $\texttt{t}$ apparaît dans la requête $\texttt{q}$; $\texttt{leads\_to}(\texttt{q}, \texttt{d})$ si la requête $\texttt{q}$ a conduit au document $\texttt{d}$ par l'intermédiaire du moteur de recherche ; $\texttt{contains}(\texttt{d}, \texttt{t})$ si le document $\texttt{d}$ contient le terme $\texttt{t}$.

De plus, à l'instar de Wang et Cohen (2007), nous créons pour toute relation $\texttt{r}$ une relation inverse $\texttt{r}^{-1}$. Nous pensons que cette modélisation bidirectionnelle des arcs offre plusieurs avantages :

- elle modélise correctement l'influence réciproque existant entre entités ;
- elle offre un modèle global où tout sommet peut influencer les autres sommets ;
- elle permet de garantir la convergence de l'algorithme de marche aléatoire vers une unique solution.

Un exemple de graphe sur le thème `Society/Folklore` de l'OpenDirectory est donné à la Figure 5.3.

Notons qu'avec le modèle décrit jusqu'alors, les mots outils qui apparaissent dans un grand nombre de documents vont obtenir une centralité très forte dans le graphe et donc un poids important par la suite. Afin d'éviter cela et de faire ressortir les termes spécialisés du domaine étudié, nous pondérons les arcs document-terme avec une mesure de spécificité (Kageura et Umino, 1996) telle que le *tfidf* ou le *log odds ratio*. Nous normalisons alors systématiquement les poids des arcs pour nous assurer du caractère stochastique du graphe avant d'appliquer notre algorithme de marche aléatoire.

5.3 Marches aléatoires

Nous appliquons à présent un algorithme de marche aléatoire sur notre graphe. Cet algorithme peut être décrit simplement en imaginant un utilisateur dont le but est de trouver des termes et documents liés à quelques termes de départ. L'utilisateur « surfe » aléatoirement sur le graphe de la manière suivante : tout d'abord, l'utilisateur choisit de manière uniforme une requête où apparaît l'un de ses termes amorces. Puis, il choisit l'un des documents trouvés par le moteur de recherche pour cette requête. Ensuite, il choisit avec une probabilité proportionnelle à la mesure de spécificité l'un des mots appartenant au document. L'utilisateur peut enfin remonter vers d'autres documents et poursuivre vers d'autres requêtes, ou redes-

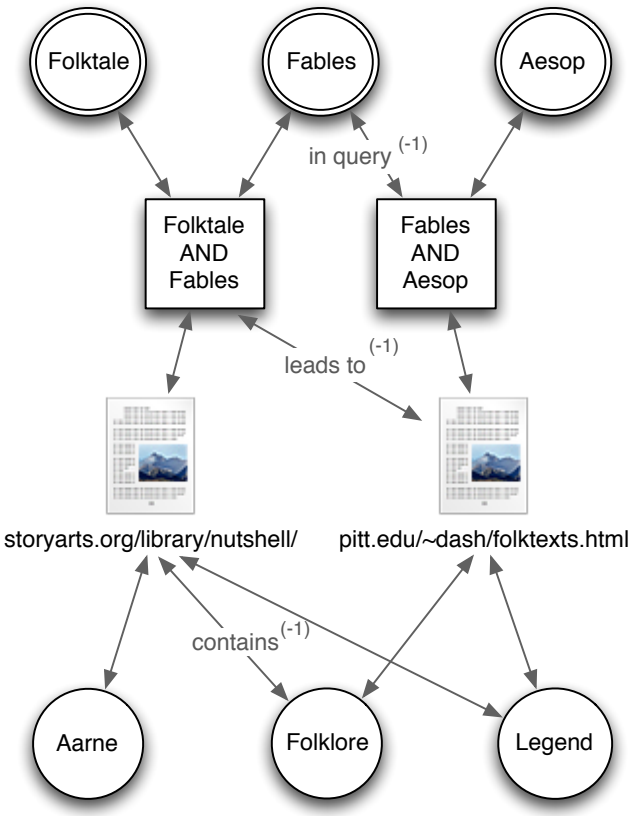


FIGURE 5.3 : Sous-graphe triparti pour le thème `Society/Folklore`.

cendre vers les mots cooccurrant avec le mot trouvé. L'utilisateur a également la possibilité à chaque étape de continuer son parcours du graphe (avec une probabilité α, dénommée facteur d'amortissement) ou de recommencer son parcours à partir d'un sommet choisi aléatoirement (avec une probabilité $1 - \alpha$, dite de « téléportation »). Enfin, de par le caractère bidirectionnel des arcs et la probabilité de téléportation $1 - \alpha$, il est prouvé qu'après un nombre d'itérations suffisamment long, l'algorithme converge vers une unique distribution de probabilités stationnaire, fournissant, pour chaque sommet, la probabilité que l'utilisateur aboutisse à ce dernier.

Formellement, la probabilité (PageRank ou PR) d'aboutir sur un sommet V_i du graphe est définie récursivement comme suit :

$$PR(V_i) = (1 - \alpha)\lambda_0(V_i) + \alpha \times \sum_{j \in In(V_i)} \frac{PR(V_j)}{|Out(V_j)|}$$

où $In(V_i)$ (resp. $Out(V_i)$) sont les prédécesseurs (resp. successeurs) du sommet V_i.

Comme nous l'avons déjà mentionné à la section 4.2 (p. 52), l'algorithme original du PageRank utilise un facteur d'amortissement α de 0.85 [3] et un vecteur de personnalisation (ou de téléportation) λ_0 uniforme. Au contraire, Richardson et Domingos (2002) ou Haveliwala (2003) ont proposé de biaiser (ou « personnaliser ») le PageRank envers certains sommets. Nous adoptons cette approche et utilisons un vecteur de personnalisation non uniforme pour que la marche aléatoire reste à proximité des termes amorces. Nous définissons le vecteur λ_0 comme suit :

$$\lambda_0 = (\sigma_0(t_0), \sigma_0(t_1), \ldots, \sigma_0(t_n))$$

$$\text{où } \sigma_0(t) = \begin{cases} 1.0 & \text{si } t \in \text{termes amorces} \\ 0.0 & \text{sinon} \end{cases}$$

Notons que l'utilisation des valeurs 0 dans le vecteur λ_0 empêche théoriquement l'irréductibilité de la chaîne de Markov et donc l'unicité de la distribution stationnaire. Toutefois, de Jager et Bradley (2009) ont récemment montré que l'unicité de la distribution stationnaire était garantie par le fait que le vecteur λ_0 soit un vecteur de probabilités et non un vecteur dense, autorisant ainsi des valeurs nulles dans le vecteur de téléportation.

Comme les arcs de notre graphe sont typés, nous modifions légèrement la définition du PageRank, à l'instar de Wang et Cohen : au moment de choisir un nœud de destination, l'utilisateur imaginaire choisit d'abord uniformément un type de relation avant de choisir un sommet de destination parmi ceux correspondant à cette relation.

Après convergence de l'algorithme de marche aléatoire, nous obtenons pour chaque entité un poids reflétant l'importance globale de l'entité dans le graphe. Nous ordonnons alors chaque type d'entité indépendamment et obtenons une liste de documents, de termes et de requêtes ordonnés par importance relative. Cet ordre peut alors être utilisé pour filtrer les documents ou termes non pertinents plus facilement.

5.4 Évaluation brute sur l'OpenDirectory

5.4.1 *Évaluation de l'ordre des documents*

Nous évaluons à présent l'apport de notre approche sur les données issues de l'OpenDirectory. Nous mesurons dans un premier temps la va-

3. Nous utilisons également cette valeur pour nos expériences. En outre, lorsque nous avons évalué l'influence de ce paramètre sur les performances, nous n'avons pas constaté de changement important.

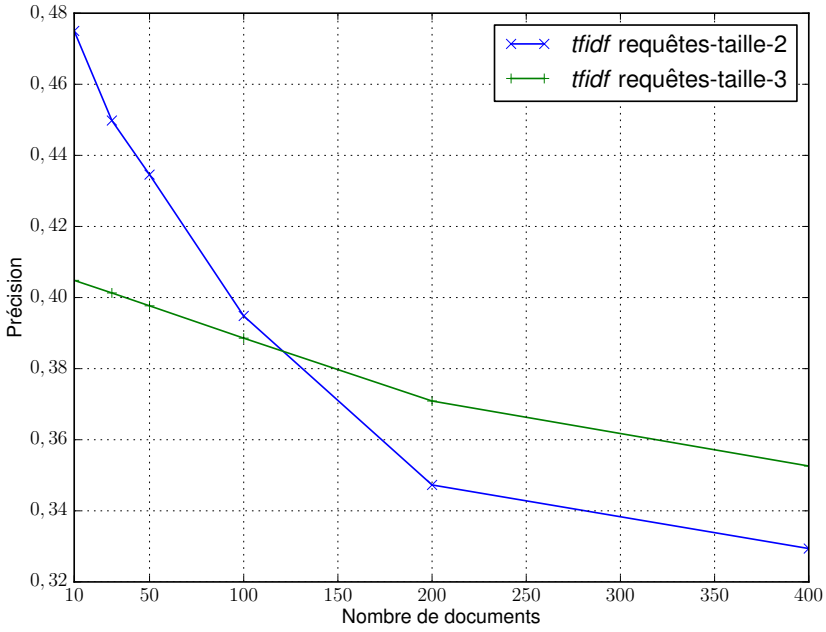


FIGURE 5.4 : Évolution de la précision à k où k est le nombre de documents retenus une fois pondérés par GrawlTCQ.

lidité de l'ordre produit par notre algorithme de marche aléatoire. Pour ce faire, nous utilisons les mesures de précision et de précision moyenne présentées aux sections 3.3 et 4.3 (p. 40 et 57).

Nous appliquons la procédure GrawlTCQ (Figure 5.1) pour chacune des catégories du second niveau de l'OpenDirectory. Comme au chapitre 3, nous utilisons les termes sélectionnés *via* la mesure du *tfidf* comme termes amorces et générons toutes les requêtes de tailles 2 et 3. De même, nous stockons pour chaque requêtes les 10 premiers documents renvoyés par le moteur de recherche. En sortie de GrawlTCQ, nous obtenons alors une liste de documents ordonnés et évaluons la précision à k obtenue pour un nombre de documents k entre 10 et 400, la valeur 400 correspondant au nombre maximum de documents. Nous présentons les résultats à la Figure 5.4. Nous constatons que la précision décroît avec l'augmentation du nombre de documents, ce qui semble indiquer que l'ordre produit est valide. Nous constatons également une chute des performances plus importante pour les requêtes composées de deux termes. Pour ces dernières, l'algorithme de marche aléatoire biaisé semble correctement positionner les documents pertinents en tête de liste, mais la forte proportion de documents non pertinents (voir § 3.4.1, p. 43) fait chuter la précision avec l'augmentation du nombre de documents.

Pour mesurer l'apport de l'ordre produit par GrawlTCQ, nous comparons les performances obtenues avec l'ensemble des documents et celles obtenues en ne conservant que les 100 documents les mieux pondérés. Notre méthode offre alors un gain en précision de $+0,065$ pour les requêtes de taille 2 et $+0,036$ pour les requêtes de taille 3 en moyenne. Une étude approfondie de ces résultats montre également que certaines catégories bénéficient de notre méthode de manière plus importante que d'autres. C'est par exemple le cas des catégories `Health/Pharmacy` (+0,28) ou `Science/Technology` (+0,21). Au contraire d'autres catégories bénéficient très peu de l'ordre produit comme `Arts/Art_History` (+0,002) ou `Arts/Humanities` (+0,006). Contrairement à nos attentes, le coefficient de corrélation de Pearson n'a montré aucune corrélation significative entre le gain obtenu par notre méthode et la pertinence thématique des corpus téléchargés.

5.4.2 *Évaluation de la sélection de termes amorces*

Nous étudions maintenant l'impact de notre approche sur la robustesse de la sélection de mots-clés au fur et à mesure des itérations. En effet, GrawlTCQ offre une méthode globale pour pondérer termes, documents et requêtes. Les probabilités produites pour l'ensemble des termes peuvent ainsi être réutilisées pour sélectionner de nouveaux termes amorces dans le cadre d'une application itérative.

Pour évaluer la sélection de termes amorces, nous proposons d'étudier l'évolution de la précision au fur et à mesure des itérations. En effet, la sélection de mauvais termes amorces devrait nous amener à de mauvais documents et faire ainsi chuter la précision. Au contraire, si notre sélection de termes est bonne, alors la dérive thématique devrait être réduite et les performances devraient décroître moins rapidement.

Après avoir réalisé une première fois cette expérience, nous avons constaté que l'algorithme de nettoyage de pages, dont le rôle est de transformer les documents HTML en textes, influait fortement sur la qualité des termes extraits. En effet, en conservant l'intégralité du texte des pages HTML, certains termes apparaissant dans les menus, publicités où les articles liés sont lourdement répétés et obtiennent une importance injustifiée. Nous avons donc dans un premier temps évalué plusieurs algorithmes de nettoyage de pages Web état de l'art (voir l'annexe A). Nous avons retenu l'algorithme BodyTextExtraction (Finn et coll., 2001) et utilisons ce dernier dans la suite de notre expérience.

Nous comparons une approche itérative standard (équivalente à l'approche BootCaT sans filtrage manuel (Baroni et Bernardini, 2004)) à notre approche GrawlTCQ. À chaque itération, chacune des méthodes choisit deux nouveaux termes qui sont ajoutés à l'ensemble des amorces déjà choisies. Comme précédemment, nous réalisons toutes les combinaisons de termes possibles de tailles 2 et 3 pour les catégories du second niveau de l'OpenDirectory. De même, nous téléchargeons les 10 meilleurs documents pour chaque requête. Nous évaluons la précision des documents récupérés par les deux méthodes au fur et à mesure des itérations. La mesure de précision ne tient pas compte de l'ordre produit par GrawlTCQ. Par conséquent, dans cette expérience, seule la qualité des termes amorces à chaque itération influe sur la qualité des corpus créés.

Pour comparer les deux méthodes dans des conditions proches, nous pondérons les arcs du graphe de GrawlTCQ par le rapport des chances (*log odds ratio*), qui est également utilisé dans BootCaT pour sélectionner les termes amorces à chaque itération. Pour éviter toute erreur numérique, nous appliquons également un lissage additif (*add-one smoothing*) dans les deux cas. Le rapport des chances permet de distinguer les termes spécialisés des termes généraux. Pour ce faire, il s'appuie sur la fréquence d'apparition des termes dans le corpus étudié (noté E) et un large corpus de référence (noté R). Formellement, il est défini comme suit :

$$\text{log odds ratio}_{t,E,R} = \log(tf_{t,E}) + \log(N_R - tf_{t,R}) - \log(tf_{t,R}) - \log(N_E - tf_{t,E})$$

où $tf_{t,C}$ fournit le nombre d'occurrences du terme t dans le corpus C, et N_C correspond au nombre total de termes dans le corpus C. Dans toutes nos expériences, nous avons utilisé le corpus *ukWac* (Ferraresi et coll., 2008), un très large corpus généraliste en langue anglaise dérivé du Web, comme corpus de référence.

Nous présentons les résultats de cette expérience à la Figure 5.5. La Figure 5.5(a) présente les résultats obtenus en partant de termes amorces initiaux sélectionnés *via* le *tfidf* alors que la Figure 5.5(b) présente ceux obtenus en partant de termes sélectionnés *via* notre critère de cohésion thématique (Chapitre 4, p. 49). Nous observons une précision plus haute tout au long des itérations lorsque les amorces initiales sont sélectionnées par le critère de cohésion thématique. De plus, nous constatons que le gain obtenu par l'utilisation de GrawlTCQ s'amenuise au fur et à mesure des itérations. Rappelons que Baroni et Bernardini (2004) appliquent BootCaT sur seulement 2 à 3 itérations. Par conséquent, nous nous concentrons plus précisément sur le gain obtenu dans les premières itérations et reportons, au Tableau 5.1, la différence entre la précision obtenue par GrawlTCQ et par BootCaT.

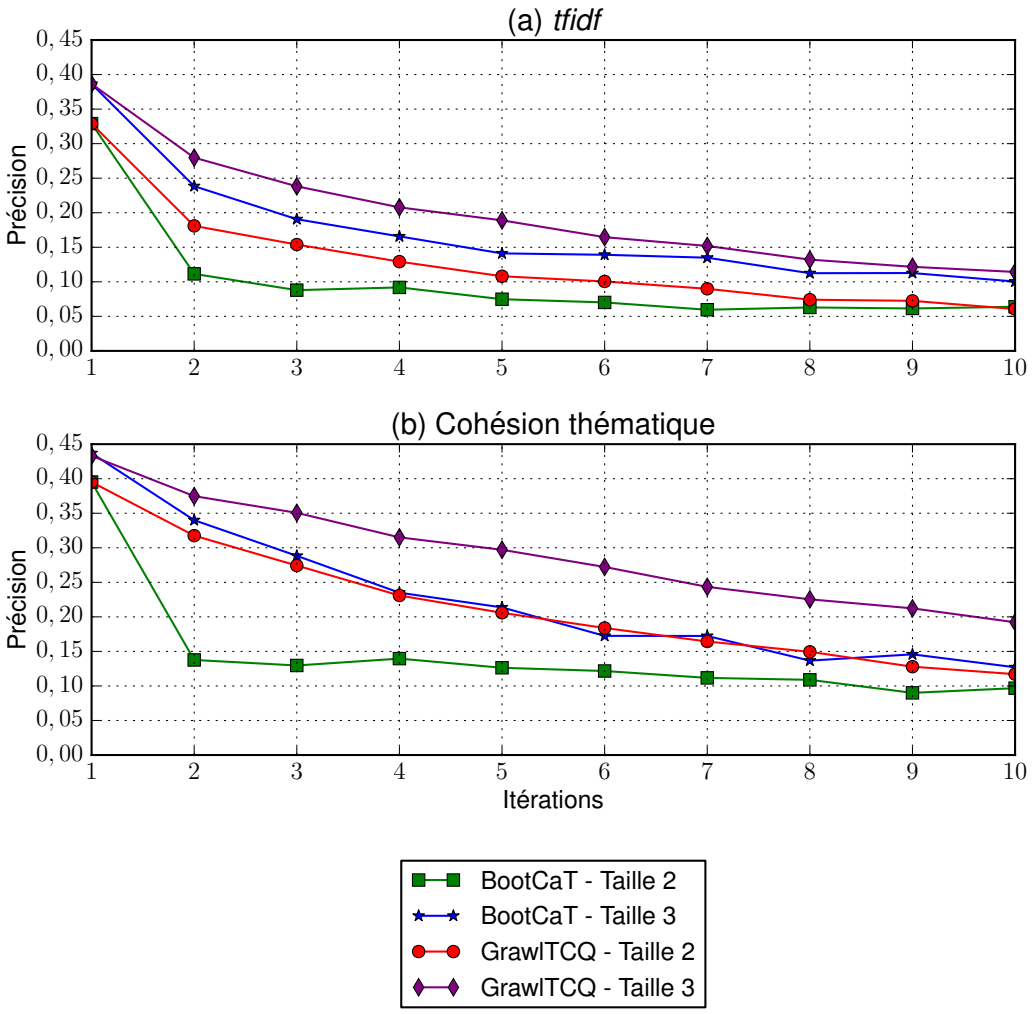


FIGURE 5.5 : Évolution de la précision des corpus constitués au fur et à mesure des itérations. La Figure (a) présente les résultats obtenus en partant de termes sélectionnés *via* le *tfidf*. La Figure (b) fait au contraire usage de la cohésion thématique (chapitre 4).

Le gain obtenu en utilisant l'algorithme GrawlTCQ est supérieur pour les requêtes composées de deux termes. Nous supposons que ces dernières renvoient de nombreux documents non pertinents et que notre algorithme de marche aléatoire biaisé permet de discriminer correctement les documents non pertinents et d'utiliser cette information pour mieux sélectionner les termes amorces de l'itération suivante. La meilleure précision est obtenue pour des requêtes composées de trois termes et utilisant à la fois GrawlTCQ et la cohésion thématique. Pour conclure, l'utilisation conjointe de nos deux traitements réduit de manière significative la dérive thématique et permet d'obtenir une diminution de la précision moins abrupte tout au long des itérations.

Amorces	Taille Req.	Gain à l'itération 2	Gain à l'itération 3
tfidf	2	+0,069 (0,329)	+0,066 (0,181)
	3	+0,041 (0,386)	+0,048 (0,280)
Cohésion thématique	2	+0,180 (0,395)	+0,145 (0,318)
	3	+0,035 (0,433)	+0,062 (0,375)

Tableau 5.1 : Gain fourni par l'algorithme GrawlTCQ en précision pour les deux premières itérations. La précision de GrawlTCQ est présentée entre parenthèses.

5.5 Évaluation sur un corpus de l'Agence France Presse

Nous avons constaté à la précédente section que GrawlTCQ permet d'ordonner les documents par pertinence thématique et de sélectionner de meilleurs termes amorces à chaque itération. Nos différentes évaluations ont montré un gain intéressant sur les données de l'OpenDirectory, ainsi qu'une différence notoire entre les performances obtenues avec les termes amorces sélectionnés *via* le *tfidf* et la cohésion thématique. De plus, nous avons constaté que la méthode de nettoyage de pages influait également sur la qualité des corpus produits. Il nous semble donc pertinent d'évaluer les performances de GrawlTCQ dans un contexte où : *(i)* les termes amorces seraient spécifiés par des experts et non extraits automatiquement de textes ; *(ii)* les documents seraient des textes « propres » ; *(iii)* les thèmes auraient une granularité plus large.

Nous avons évalué GrawlTCQ pour la création de corpus d'actualités. Pour ce faire, nous avons utilisé un corpus en anglais de 57 441 dépêches de l'AFP (Agence France Presse) écrites entre le 1er janvier et le 31 mars 2010 [4]. Les dépêches sont stockées au format NewsML [5] et annotées d'une ou plusieurs catégories de l'arbre thématique IPTC [6] (International Press Telecommunications Council). De plus, plusieurs métadonnées accompagnent chaque document, notamment des mots-clés entrés manuellement par les auteurs des dépêches. Nous avons considéré les 17 catégories de premier niveau de l'arbre IPTC. Comme certaines catégories contenaient trop peu de documents, nous n'avons conservé que les documents des 10 catégories les plus larges, présentées au Tableau 5.2. Nous avons ensuite indexé les documents de ces 10 catégories dans le moteur de recherche Lu-

4. Nous tenons à remercier l'AFP de nous avoir autorisé à utiliser ces données dans le cadre de nos recherches.
5. http://www.iptc.org/site/News_Exchange_Formats/NewsML_1/
6. http://www.iptc.org

Id	Category	#docs
01	Arts, culture, et spectacles	3074
02	Police et justice	5675
03	Désastres et accidents	4602
04	Économie et finances	13321
08	Gens animaux insolite	1300
11	Politique	17848
12	Religion et croyance	1491
14	Société	1764
15	Sport	15089
16	Guerres et conflits	8589

Tableau 5.2 : Distribution des 10 catégories IPTC les plus peuplées dans le corpus de l'Agence France Presse.

cene en suivant le même procédé que celui décrit au Chapitre 3 (p. 35).

Nous comparons les algorithmes BootCaT et GrawlTCQ sur une tâche identique : créer des corpus de 50, 100, 300, 500 et 1000 documents pour chaque catégorie, indépendamment du nombre d'itérations réalisées.

Afin de sélectionner les termes amorces initiaux, nous nous appuyons sur les métadonnées du corpus. Nous appliquons la mesure de spécificité *Weirdness* (Ahmad et coll., 1999) à l'ensemble des mots-clés du corpus, renseignés par des experts du domaine (journalistes). Cette mesure est calculée comme suit :

$$\text{weirdness}_{t,E,R} = \frac{tf_{t,E}/N_E}{tf_{t,R}/N_R}$$

où R est un corpus composé des mots-clés de l'ensemble des catégories et E correspond aux mots-clés d'une catégorie. Cette mesure permet d'extraire des termes amorces pertinents, mais pas nécessairement exclusifs à une catégorie. Par exemple, les 10 mots-clés les mieux classés pour la catégorie « Économie et finances » sont : *economics*, *summary*, *rate*, *opec*, *distress*, *recession*, *zain*, *jal*, *gold*, et *spyker*.

Nous fixons ensuite un ensemble de paramètres : à chaque itération, les 10 meilleurs termes sont sélectionnés comme amorces (qu'il s'agisse d'un terme amorce initial ou nouveau). Les requêtes sont composées de paires de termes et toutes les combinaisons possibles sont réalisées. Enfin, pour

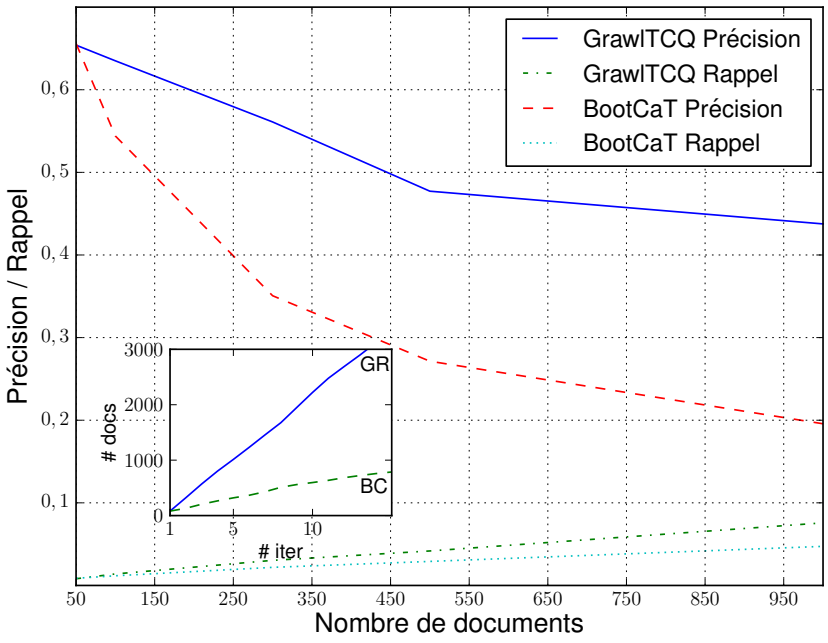


FIGURE 5.6 : Précision et rappel moyens pour 50, 100, 300, 500 and 1000 documents (insert : nombre moyen de documents / nombre d'itérations)

chaque requête, les 10 meilleurs documents sont conservés.

Nous avons calculé la précision et le rappel des deux algorithmes sur les 10 catégories. Puis nous avons calculé la macro-moyenne de ces mesures sur l'ensemble des catégories. Comme nous pouvons le constater à la Figure 5.6, GrawlTCQ est beaucoup plus robuste à la dérive des amorces/thèmes et surpasse BootCaT de 25 % en précision pour 1000 documents récupérés. Les détails de ces résultats pour chaque catégorie sont présentés au Tableau 5.3 et confirment la pertinence de notre approche. Étonnamment, BootCaT et GrawlTCQ ont une précision très faible pour la catégorie n° 14 « Société ». Nous pensons que les documents de cette catégorie sont relativement ambigus et que la terminologie de ce domaine est difficile à extraire.

Nous avons également tracé l'évolution du nombre de documents en fonction du nombre d'itérations en insert de la Figure 5.6. La courbe montre clairement que le nombre de documents augmente plus rapidement pour l'algorithme GrawlTCQ. Nous pensons que cela est dû à la pertinence des termes amorces sélectionnés : les requêtes formulées par l'algorithme GrawlTCQ mènent à de nombreux documents alors que celles formulées par BootCaT mènent à peu ou pas de documents. En outre, GrawlTCQ permet de récupérer plus rapidement plus de documents, mais la précision moyenne des corpus reste également supérieure à celle de BootCaT

CatId	P@50		P@100		P@300		P@500		P@1000	
	GR	BC	GR	BC	GR	BC	GR	BC	GR	BC
01	**0,58**	0,50	**0,57**	0,30	**0,43**	0,12	**0,35**	0,08	**0,23**	0,05
02	0,44	**0,60**	**0,45**	0,33	**0,46**	0,17	**0,44**	0,10	**0,34**	0,07
03	0,82	0,82	**0,99**	0,81	**0,89**	0,41	**0,66**	0,26	**0,54**	0,14
04	**0,86**	0,80	0,82	**0,85**	**0,84**	0,55	**0,78**	0,34	**0,79**	0,19
08	0,72	0,72	0,44	**0,48**	0,23	**0,42**	0,17	**0,40**	0,20	**0,39**
11	0,76	**0,78**	0,79	**0,81**	**0,87**	0,71	0,57	**0,64**	**0,57**	0,56
12	0,46	**0,54**	**0,35**	0,27	**0,20**	0,10	**0,17**	0,06	**0,15**	0,03
14	0,08	**0,24**	**0,13**	0,10	**0,06**	0,04	**0,04**	0,02	**0,04**	0,02
15	1.0	1.0	1.0	1.0	**0,92**	0,78	**0,87**	0,67	**0,81**	0,39
16	**0,82**	0,56	**0,81**	0,49	**0,71**	0,21	**0,72**	0,15	**0,70**	0,13

Tableau 5.3 : Précision des algorithmes GrawlTCQ (GR) et BootCaT (BC) par catégorie.

ce qui montre que les termes amorces choisis sont, en même temps, plus prolifiques et plus pertinents.

5.6 Bilan

Dans ce chapitre, nous nous sommes intéressé au filtrage de corpus créés par le biais de requêtes à un moteur de recherche. Nous avons défini GrawlTCQ, un algorithme fondé sur un modèle de graphe incluant termes, requêtes et documents. Nous avons appliqué un algorithme de marche aléatoire sur ce graphe afin d'attribuer un score d'importance globale à l'ensemble des sommets simultanément. Ce score permet alors d'ordonner les documents et termes par ordre décroissant d'importance. L'ordre des documents peut ensuite être utilisé pour aider un utilisateur à filtrer les documents rapatriés. Par ailleurs, l'algorithme ordonne également les termes des documents, permettant de sélectionner de nouveaux termes amorces plus pertinents lorsque la procédure de recherche orientée est appliquée de manière itérative.

Nous avons démontré l'intérêt de notre méthode par le biais de deux expériences reposant sur deux jeux de données distincts : l'OpenDirectory et un corpus de dépêches d'actualités de l'Agence France Presse. Nos résultats montrent que GrawlTCQ peut permettre d'améliorer la qualité des corpus produits en ordonnant et en ne conservant que les documents les mieux classés. De plus, GrawlTCQ permet d'améliorer de manière notable

la sélection de termes amorces pour la recherche orientée itérative.

Nous considérons plusieurs pistes comme travaux futurs. Tout d'abord, nous voudrions évaluer l'apport de GrawlTCQ pour l'extraction terminologique. En effet, notre méthode assigne à chaque terme un score sur lequel il semble intéressant de s'appuyer pour constituer des terminologies. Des expériences préliminaires sur trois catégories (Astronomie, Football et Gastronomie) ont montré un gain intéressant sur la précision des termes extraits, par rapport à la mesure du *tfidf* (Wermter et Hahn, 2006). Toutefois, ces expériences doivent être reproduites dans un cadre plus rigoureux et à plus large échelle (terminologies de référence, nombreux thèmes) pour pouvoir en tirer de réelles conclusions.

La méthode que nous avons définie permet d'ordonner les documents par importance. La sélection des documents à partir de cet ordre est ensuite réalisée manuellement ou automatiquement en choisissant un nombre de documents fixe. Il semble prometteur de tenter de déterminer automatiquement un seuil au-delà duquel filtrer les documents. Nous voudrions également approfondir nos recherches concernant le modèle de graphe. En effet, Nie et coll. (2005) ont proposé de pondérer les relations des arcs en s'appuyant sur un petit ensemble d'éléments annotés. Il serait intéressant d'évaluer l'impact de cette approche sur notre modèle en annotant un petit ensemble de documents.

Enfin, nous avons présenté dans ce chapitre une méthode non supervisée dont l'objectif est d'ordonner les termes et documents par pertinence. Pour améliorer le filtrage des documents, une solution alternative serait d'opter pour une méthode supervisée, à l'instar des travaux de Ghani et coll. (2005). Contrairement à GrawlTCQ, une telle approche ne permet pas de traiter simultanément terminologie et corpus, mais devrait néanmoins nous permettre d'obtenir des corpus moins bruités. La différence majeure entre une approche supervisée et GrawlTCQ réside dans le fait qu'il est nécessaire de disposer d'un corpus d'apprentissage composé d'exemples positifs et négatifs. Afin d'éviter la collecte d'exemples négatifs, nous pourrions nous tourner vers des méthodes d'apprentissage à partir d'exemples positifs (*One-class SVM* (Manevitz et Yousef, 2002)) ou d'exemples positifs et non étiquetés (*PU-Learning* (Liu et coll., 2003)) pour obtenir un catégoriseur thématique. Les exemples positifs, documents pertinents pour le thème, seraient alors sélectionnés manuellement après une première itération de la méthode, ou encore à partir d'un corpus existant (l'OpenDirectory par exemple). De plus, en augmentant encore légèrement l'effort d'annotation, le catégoriseur thématique obtenu par cette approche pourrait être affiné par la suite par apprentissage actif (Settles, 2010).

6 EXPLORATION ORIENTÉE DU WEB

Comme nous l'avons constaté, les moteurs de recherche Web offrent un point d'entrée simple et efficace à un très large ensemble de documents du Web. Toutefois, leur utilisation impose plusieurs limites : des critères de sélection et de tri inconnus, un nombre de résultats limité, une durée de vie incertaine ou encore une spécification du besoin réduite (voir la discussion complète à la section 2.3, p. 20).

Pour pallier ces limites, nous nous orientons à présent vers une approche de collecte alternative ne nécessitant pas de moteur de recherche existant : le *crawling* orienté ou *focused crawling* (Chakrabarti et coll., 1999). Cette approche démarre à partir de pages amorces et parcourt le Web au travers des hyperliens, à l'instar des robots d'indexation des moteurs de recherche Web. Cependant, à la différence de ces derniers qui explorent le Web en largeur et indexent plusieurs milliards de pages Web (19.2 milliards de pages Web pour Yahoo ! en 2005 [1], 120 milliards pour Cuil en 2010 [2]), l'exploration orientée dirige son parcours vers les pages les plus pertinentes pour son domaine. Par conséquent, la pierre angulaire des *crawlers* orientés est la priorisation des URL à télécharger, ou en d'autres termes, l'ordonnancement de la *frontière de crawl*.

Ce chapitre s'articule en deux parties. Tout d'abord, nous décrivons le *crawler* orienté *Babouk* (de Groc, 2011; de Groc et coll., 2011a) que nous avons réalisé dans le cadre du projet de recherche TTC et qui a également été utilisé pour le projet Metricc (Alonso et coll., 2012). Puis, nous nous focalisons sur les stratégies d'exploration et proposons une méthodologie originale permettant d'apprendre une fonction d'ordonnancement pour ordonner la frontière de *crawl*.

6.1 Babouk : exploration orientée du Web

Dans cette section, nous décrivons l'implémentation singulière de Babouk qui nous permet de répondre aux contraintes imposées par les projets de recherche TTC et Metricc. Puis, nous décrivons les stratégies de base que nous utilisons pour orienter le *crawl*.

1. http://www.ysearchblog.com/archives/000172.html
2. http://www.cuil.com/

6.1.1 *Implémentation pour un passage à l'échelle*

CONTEXTE ET CONTRAINTES

Babouk a été réalisé dans le cadre du projet européen TTC. Ce projet multi partenaire doit permettre à plusieurs utilisateurs de constituer simultanément de larges corpus à partir du Web. La complexité technique est donc double : d'une part, le *crawler* doit être aussi performant que possible. D'autre part, plusieurs instances de ce *crawler* peuvent être lancées simultanément par plusieurs utilisateurs. Pour conserver des performances acceptables, il semble donc nécessaire d'implémenter un crawler distribué pouvant être répliqué à grande échelle.

Nous avons considéré plusieurs *crawlers* Open Source comme base pour notre *crawler* orienté :

- Bixo [3], une boîte à outils permettant le téléchargement et le traitement de pages Web de manière distribuée.
- Heritrix (Burner, 1997), le crawler du site *Internet Archive* [4] qui met l'accent sur l'archivage de pages Web.
- Nutch [5] (Khare et coll., 2004), un *crawler* distribué basé sur la plateforme Hadoop [6].

Toutes ces options fournissent les briques de base pour la réalisation de notre *crawler* orienté telles que la détection d'encodage, la gestion de différents formats de fichiers (PDF, DOC, texte) et la détection de langue du document. De plus, ils gèrent également la parallélisation et la politesse envers les serveurs Web de manière native.

L'ÈRE DU BIG DATA

Les années 2000 ont certainement été celles de l'avènement du *Big Data*. La montée en puissance des méthodes empiriques depuis les années 90 ainsi que l'augmentation constante de la quantité de données disponibles, notamment *via* Internet, a amené les milieux industriels et académiques à acquérir, manipuler et traiter toujours plus de données. Ainsi, Banko et Brill (2001) ont étudié une tâche typique de classification en traitement automatique des langues et ont conclu, dans un article devenu célèbre, qu'obtenir plus de données était plus important que de définir de meilleurs algorithmes. Brants et coll. (2007) ou Halevy et coll. (2009) rapportent des résultats similaires appliqués à d'autres domaines. Mercer avait su, dès

3. http://openbixo.org
4. http://archive.org/
5. http://nutch.apache.org
6. http://hadoop.apache.org

1985, résumer les conclusions de ces derniers en une phrase : *There is no data like more data*[7].

Face à cette montée en volume des données, l'industrie s'est tournée vers l'*évolutivité horizontale* (*scale-out* ou *horizontal scaling*) (Michael et coll., 2007) : la distribution des traitements sur une grappe de machines « bon marché ». Cette tendance a atteint son paroxysme avec l'introduction du paradigme MapReduce (Dean et Ghemawat, 2004) par Google, repris par la suite par le projet Open Source Hadoop. Les professionnels du *Cloud computing*, tels que le précurseur *Amazon Web Services*, proposent de nos jours des formules permettant l'allocation à la demande de machines disposant de la plateforme Hadoop[8]. Il semble donc opportun de nous appuyer sur cette plateforme et les outils du *Cloud computing* permettant l'ajout de nouvelles machines de manière souple, en fonction de l'affluence et de l'utilisation de notre outil.

NUTCH

Pour réaliser notre *crawler* orienté, nous avons choisi de nous appuyer sur le *crawler* Open Source Nutch, lui-même basé sur la plateforme Hadoop. Nutch opère un parcours en largeur du Web et a déjà indexé des collections avoisinant les 100 millions de pages (Khare et coll., 2004). Bien que ce nombre semble trop faible pour la création d'un index global du Web, il semble tout à fait pertinent pour la réalisation d'un moteur de recherche vertical.

La procédure de *crawling* de Nutch se décompose en une succession de tâches itérées jusqu'à ce qu'un critère d'arrêt soit atteint (Figure 6.1). Tout d'abord les URL amorces sont injectées dans la frontière de *crawl* (*CrawlDB*). Puis, la boucle de *crawling* démarre : un ensemble d'URL est sélectionné puis distribué sur l'ensemble des machines pour être téléchargé puis analysé. Les nouvelles URL sont alors ajoutées à la frontière de *crawl*, initiant l'itération de *crawling* suivante.

Nutch est pourvu d'un grand nombre de points d'entrée (ou *plug-ins*), permettant la personnalisation et l'ajout de post-traitements tels que :
- Des *plug-ins* pour ordonner la frontière de *crawl* (*ScoringFilter*)
- Des *plug-ins* d'extraction de contenu (*Parser*)
- Des filtres sur les documents (*ParseFilter*)
- Des filtres sur les URL (*URLParseFilter*)

7. Une traduction possible serait : « Il n'y a pas meilleures données que plus de données. »
8. http://aws.amazon.com/ec2/

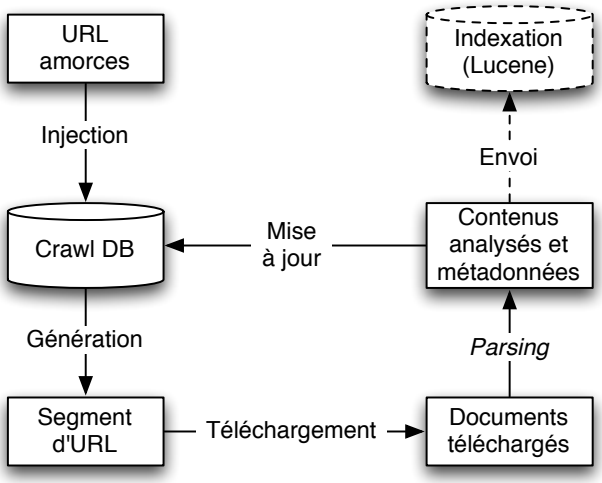


FIGURE 6.1 : Procédure de *crawling* de Nutch

Enfin, Nutch s'appuie sur une base de données distribuée pour stocker les informations concernant les pages téléchargées et la frontière de *crawl*. Cette dernière peut être le système de fichiers HDFS [9] ou une base de données NoSQL ou SQL, telle qu'Apache Cassandra [10] ou MySQL [11].

PERSONNALISATION DE NUTCH POUR LE CRAWLING ORIENTÉ

Nous nous sommes appuyés sur le système de *plug-ins* de Nutch afin d'orienter son parcours. Pour ce faire, nous avons :

- Implémenté un *plug-in* permettant de modifier l'ordre de la frontière de *crawl* (*ScoringFilter*).
- Implémenté plusieurs *plug-ins* permettant d'appliquer nos analyseurs sur les pages téléchargées (*Parse*, *ParseFilter*).

Dans Babouk, l'ordonnancement de la frontière de *crawl* (*ScoringFilter*) est réalisée par l'une des deux méthodes suivantes :

1. Une version simplifiée du critère OPIC (*On-Line Page Importance Computation*) de Abiteboul et coll. (2003) décrite dans Baeza-Yates et coll. (2005).
2. Une variante de la stratégie Shark-Search de Hersovici et coll. (1998) qui intègre de manière heuristique la pertinence thématique des pages, des textes d'ancrage et des contextes des liens. Là où les auteurs jugent une page pertinente si sa similarité par rapport à quelques mots clés est supérieure à zéro (Hersovici et coll., 1998, fig. 2, item 1),

9. http://hadoop.apache.org/hdfs
10. http://cassandra.apache.org/
11. http://www.mysql.fr/products/cluster/

nous utilisons un catégoriseur thématique.

La stratégie Shark-Search et beaucoup d'autres (Rennie et McCallum, 1999; Chakrabarti et coll., 2002; Diligenti et coll., 2000) s'appuient sur un catégoriseur thématique permettant de prédire la pertinence d'une page par rapport au thème du *crawl*. Afin de permettre aux utilisateurs de spécifier eux-mêmes leurs thèmes, nous nous sommes orienté vers une méthode à base d'exemples positifs, fournis par l'utilisateur, et d'exemples non étiquetés sélectionnés automatiquement. Les détails de cette méthode sont présentés ci-après.

Pour les besoins du projet de recherche TTC, nous avons également ajouté un ensemble de filtres [12] permettant aux utilisateurs d'améliorer la qualité des corpus textuels :

- Extraction du « contenu informatif » des pages Web *via* l'algorithme BodyTextExtraction (BTE) (Finn et coll., 2001).
- Filtrage de documents : format de fichier, langue, nombre de tokens, taille des fichiers (Fletcher, 2004).
- Filtrage de certaines variantes de *spam* (Gyöngyi et Garcia-Molina, 2005) : Arbre de décision C4.5 basé sur des traits extraits du contenu des pages (Fetterly et coll., 2004; Ntoulas et coll., 2006).
- Filtrage de certains domaines ou URL : expressions régulières.

Enfin, le *crawling* étant un processus récursif, il nécessite un critère d'arrêt pour se terminer. Dans le cadre d'un *crawl* orienté simple qui s'arrête à toute page non pertinente, un premier critère d'arrêt est le fait de ne plus trouver de document pertinent dans la frontière de *crawl*. Pour fournir plus de contrôle aux utilisateurs, nous avons également implémenté des critères d'arrêt supplémentaires tels qu'un nombre maximal de tokens ou de documents à télécharger, ainsi qu'une profondeur ou une durée de *crawl* maximale.

Entraînement d'un catégoriseur à partir d'exemples positifs et non étiquetés

Afin d'évaluer la pertinence thématique d'une page Web, nous souhaitons nous appuyer sur un classifieur supervisé. Pour ce faire, nous ne disposons que de pages pertinentes sélectionnées par un utilisateur. Nous avons donc envisagé de construire un catégoriseur uniquement à partir d'exemples positifs tels que les machines à vecteurs de support à une classe (*One-Class SVM*) (Manevitz et Yousef, 2002). Cependant, il a été montré à plusieurs reprises (Liu et coll., 2003; Yu et coll., 2004) que l'ajout

12. Ces filtres sont désactivés par défaut lors de nos expériences.

d'exemples non étiquetés (*PU-Learning*) permettait d'obtenir de meilleures performances. Notons que bien qu'il soit tentant de considérer l'ensemble des exemples non étiquetés comme exemples négatifs, en raison de la faible proportion d'exemples positifs sur le Web, Yu et coll. (2004) ont montré qu'une approche de type *PU-Learning* offrait des performances supérieures, notamment en raison de la sensibilité du classifieur SVM aux erreurs d'apprentissage.

Pour générer aléatoirement des exemples non étiquetés, nous avons considéré les méthodes de Bharat et Broder (1998) et Bar-Yossef et Gurevich (2006). La première méthode ne correspond pas à notre cadre applicatif, car elle génère aléatoirement des requêtes à partir d'un lexique des mots apparaissant sur le Web ainsi qu'une estimation de leur fréquence d'apparition (calculée sur l'intégralité du *Yahoo ! Directory*). Au contraire, Bar-Yossef et Gurevich proposent une approche fondée sur un algorithme de marche aléatoire qui ne nécessite aucune ressource. Comme cette méthode a un coût d'exécution élevé, il convient toutefois de précalculer un large ensemble de pages choisies aléatoirement et d'utiliser cet ensemble par la suite. Enfin, une méthode alternative et beaucoup plus efficace est d'utiliser une copie locale d'un large répertoire de sites Internet (Yahoo ! Directory, OpenDirectory) et de la considérer comme représentative du Web. Nous optons pour cette méthode dans la suite de nos expériences.

PEBL (Positive Example Based Learning) (Yu et coll., 2004) est une méta-méthode pour la création de catégoriseurs de pages Web à partir d'exemples positifs et non étiquetés. L'algorithme est composé de deux phases :

- *Mapping* : les documents les « plus négatifs » sont annotés comme exemples négatifs *via* un classifieur faible.
- *Convergence* : un classifieur SVM est entraîné à partir des exemples positifs et négatifs puis est utilisé pour classer les exemples restants. Cette procédure est répétée jusqu'à ce qu'il ne reste que des exemples inclassables (ambigus).

La procédure est résumée en cinq étapes à la Figure 6.2. Notons qu'en supprimant les exemples ambigus, PEBL va à l'encontre d'une hypothèse de l'apprentissage actif selon laquelle les exemples les plus ambigus sont les plus intéressants. Toutefois, la suppression des exemples ambigus permet également une meilleure généralisation.

DÉPLOIEMENT

Notre objectif premier est la mise en place d'une grappe de *crawling* sur le *cloud*. Pour ce faire, il est nécessaire de disposer d'outils permettant de rapidement déployer le *crawler*. Nous avons donc implémenté des pro-

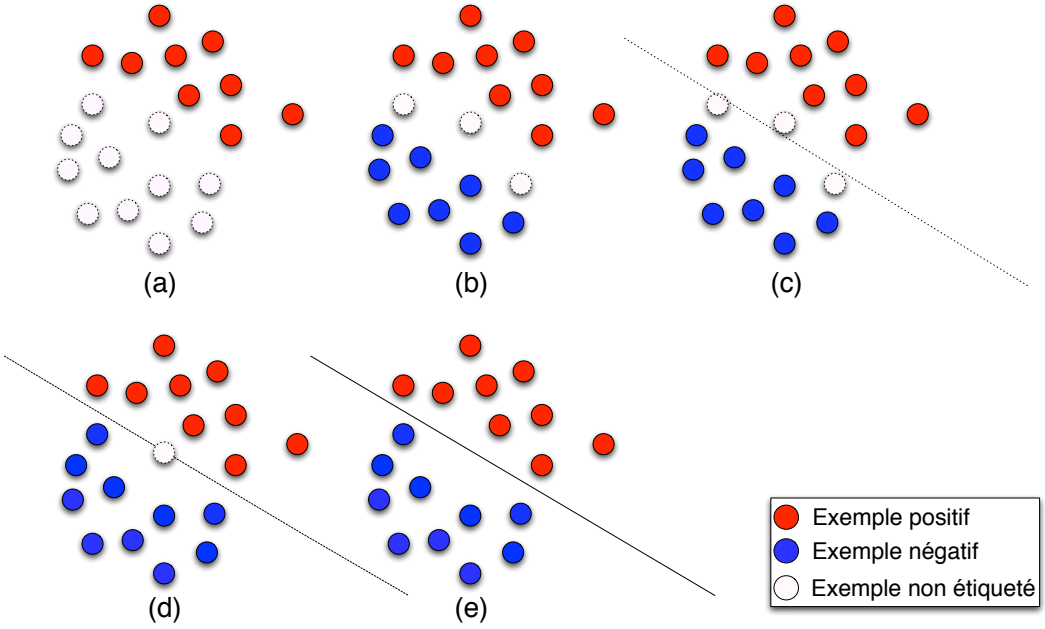


FIGURE 6.2 : Synthèse de l'algorithme *Mapping-Convergence* : (a) Ensemble initial (b) Sélection des exemples les « plus négatifs » (c) Apprentissage d'un premier séparateur linéaire (d) Classification des exemples non étiquetés restants et apprentissage d'un second séparateur (e) Suppression des exemples ambigus restants.

cédures de déploiement automatiques, permettant d'installer et de lancer très rapidement le *crawler* sur une nouvelle machine.

INTERFACE ET UTILISATION

Nous avons réalisé une interface en ligne permettant de lancer, arrêter et suivre les *crawls* en temps réel (traces). L'interface est stockée sur un serveur distant et communique avec les différentes machines du cluster de crawl par envoi de messages. Une capture d'écran de l'interface est présentée à la Figure 6.3.

6.1.2 *Synthèse*

Babouk est un *crawler* de qualité industrielle, robuste et passant à l'échelle. Au 1er août 2012, plus de 60 utilisateurs lancent périodiquement des *crawls* dans l'une des sept langues supportées : allemand, anglais, espagnol, français, letton, russe et chinois. Depuis sa création en décembre 2010, plus de 500 *crawls* ont été lancés.

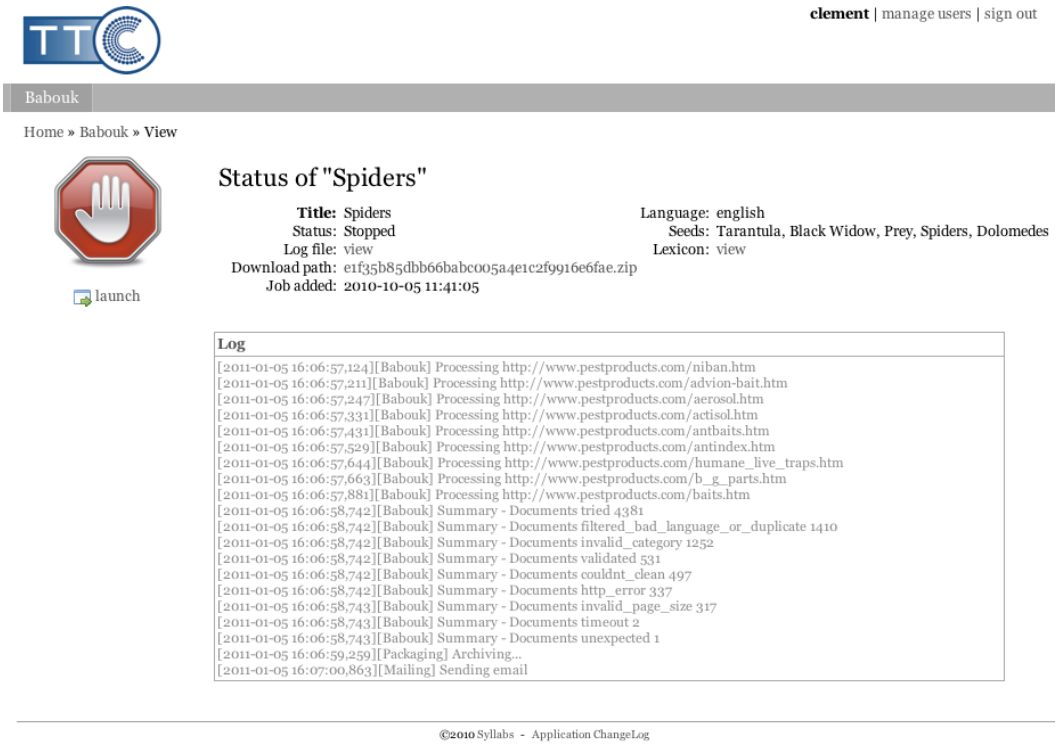


FIGURE 6.3 : Interface Web de Babouk : page de contrôle d'un *crawl*

Babouk représente également une plateforme qui va nous permettre de mener nos recherches sur le *crawling* orienté. Les stratégies d'ordonnancement OPIC et Shark-Search représentent deux approches état de l'art auxquelles nous comparer. L'une des conclusions au regard de notre état de l'art (§ 2.3.2, p. 23) et de l'implémentation de ces stratégies est notamment qu'il est nécessaire de prendre en compte et fusionner un grand nombre d'indices afin de fournir une bonne fonction d'ordonnancement : pertinence du contexte du lien, du texte d'ancrage, de l'URL, de la page parente, de l'historique de *crawl*, etc. Ce travail est souvent réalisé au travers de combinaisons linéaires dont les poids sont ajustés sur des corpus de validation (Hersovici et coll., 1998; Pant et Srinivasan, 2006). Dans la section suivante, nous proposons d'apprendre à combiner tous ces indices automatiquement dans une fonction d'ordonnancement indépendante du thème à l'aide d'algorithmes d'apprentissage de fonctions d'ordonnancement. Nous pensons que cette approche s'intègre tout à fait dans le fonctionnement par lot de Nutch et offrira une meilleure pertinence sans réduire les performances du *crawler*.

6.2 Apprendre à ordonner la frontière de *crawl*

La dernière décennie de recherches en recherche d'information a été marquée par l'avènement de nouvelles méthodes d'ordonnancement à l'intersection de l'apprentissage automatique, de la recherche d'information et du traitement des langues. Ces méthodes font usage de modèles statistiques pour apprendre à ordonner les pages Web par pertinence (apprentissage de fonctions d'ordonnancement ou *Learning to Rank*). Bien que ce thème de recherche ait été principalement porté par l'ensemble des moteurs de recherche industriels en raison des retombées économiques de ce secteur (Liu, 2009, chap. 1, p. 3), l'apprentissage de fonctions d'ordonnancement est aujourd'hui appliqué dans de nombreux domaines tels que la traduction automatique (Duh et Kirchhoff, 2008), les systèmes de recommandation (Lv et coll., 2011), l'analyse de sentiment (Pang et Lee, 2005), l'analyse de l'importance des pages Web (Richardson et coll., 2006) ou l'extraction de termes clés (Jiang et coll., 2009).

Nous proposons d'apprendre une fonction d'ordonnancement pour ordonner la *frontière de crawl*. Tout comme les moteurs de recherche généralisent des fonctions d'ordonnancement à partir de requêtes annotées, nous proposons d'apprendre une fonction d'ordonnancement à partir de *crawls* thématiques annotés pour ensuite l'appliquer à de nouveaux thèmes. Par analogie avec la recherche d'information, nous considérons que le Web est notre collection documentaire, que nos thèmes sont les requêtes et que les URL (et la pertinence du document pointé) sont les documents annotés.

L'apprentissage de fonctions d'ordonnancement offre de multiples avantages, et permet notamment de : *(i)* apprendre l'intérêt de chaque trait plutôt que de l'ajuster manuellement ; *(ii)* prendre en compte l'effet tunnel ; *(iii)* intégrer de nombreux indices (différents contextes pour les liens (Pant et Srinivasan, 2006), différentes mesures de similarités, . . .). À titre d'exemple, l'algorithme *Shark Search* (Hersovici et coll., 1998) utilise une heuristique pour intégrer la pertinence du contexte d'un lien et son texte d'ancrage. Cette tâche peut être réalisée automatiquement par l'usage d'un algorithme d'apprentissage de fonctions d'ordonnancement. De même, Pant et Srinivasan (2006) ont montré qu'il était intéressant d'utiliser simultanément le contenu complet de la page et le contexte des liens, ce que permet de faire notre approche.

En contrepartie de ces avantages, il est nécessaire de disposer d'un corpus annoté, implicitement (traces/clics utilisateurs) ou explicitement (notes), pour apprendre une fonction d'ordre. Nous faisons l'hypothèse que nous pouvons créer à partir de l'OpenDirectory des catégoriseurs suf-

fisamment performants pour générer des annotations sur des corpus de *crawl* existants. Notons que cette approche est analogue à celle de Chakrabarti et coll. (2002) qui utilisent également les annotations du catégoriseur thématique pour évaluer les performances de ses *crawls*.

6.2.1 *Apprentissage de fonctions d'ordonnancement à partir de données de crawl annotées*

Notre premier objectif est de générer des données d'entraînement sur différents thèmes que nous allons annoter par la suite. Nous avons considéré trois sources de corpus :

1. Des données de *crawl* réalisés par Babouk ;
2. Un large *crawl* du Web (tel que le corpus ClueWeb2009) ;
3. De nouvelles données de *crawl* réalisés spécifiquement pour cette tâche.

Nous n'optons pas pour la première solution qui ne présentera pas forcément beaucoup d'exemples négatifs ou de liens tunnels. La seconde solution offre l'avantage de fournir un graphe de liens relativement complet, et donc utile pour la prise en compte de l'effet tunnel, mais est beaucoup plus complexe à mettre en œuvre. Nous choisissons la dernière solution qui nous permet de contrôler la quantité de données à traiter tout en nous assurant que notre corpus dispose des caractéristiques souhaitées.

Partant des différents graphes de *crawls* G_i, nous devons les transformer en une liste de liens L_i (partiellement) ordonnée en fonction de leur ordre de visite optimale Y_i. Nous extrairons ensuite, pour chaque lien, un ensemble de descripteurs (ou traits) X_i. Les paires (X_i, Y_i) seront enfin fournies en entrée de l'algorithme qui apprendra une fonction d'ordonnancement f. Cette procédure est schématisée à la Figure 6.4.

ANNOTATION DU GRAPHE DE CRAWL

Nous avons fait l'hypothèse que nous pouvions générer automatiquement des informations de catégorisation en entraînant des catégoriseurs thématiques sur l'OpenDirectory. Bien que ce type d'approche soit couramment utilisé pour l'exploration orientée, peu d'articles mentionnent les performances de tels classifieurs sur les thèmes traités et la plupart se contentent d'évaluer ces derniers par la tâche (Pant et Srinivasan, 2005). Nous évaluons dans un premier temps les performances d'un classifieur supervisé par validation croisée sur les catégories de second niveau de l'OpenDirectory.

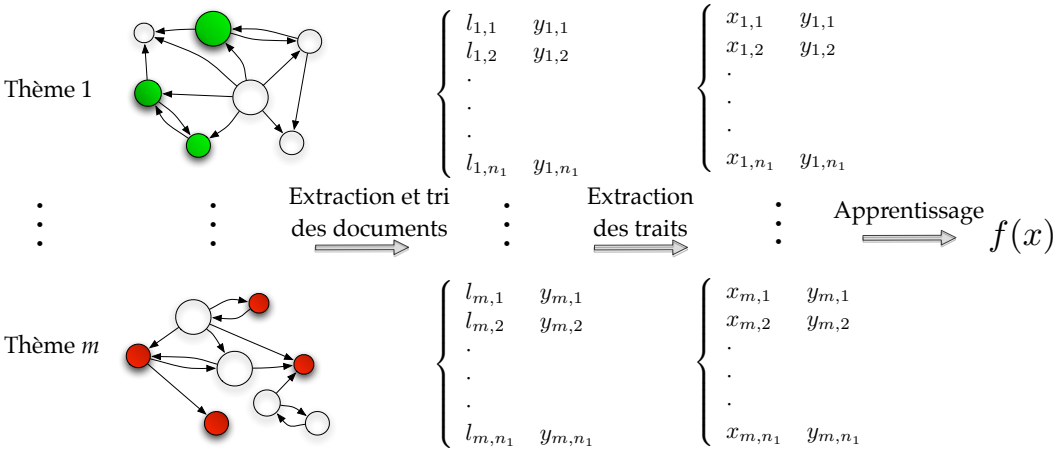


FIGURE 6.4 : Procédure d'apprentissage de fonctions d'ordonnancement.

Nous comparons deux approches : une approche purement textuelle (Sebastiani et Ricerche, 2002) et une approche tenant compte de la structure des pages Web (Choi et Yao, 2005). Pour le catégoriseur fondé sur le texte (*TextCat*), nous supprimons simplement les balises HTML des documents et représentons le texte à l'aide d'une approche sac de mots, d'une transformation *tfidf* et d'une normalisation cosinus. Pour le catégoriseur tenant compte de la structure des pages Web (*WebPageCat*), nous créons plusieurs sacs de mots distincts pour différents éléments de la page Web : le titre de la page (contenu de la balise *title*), le corps de la page (*body*), les textes d'ancrage (*a*), les méta-informations (*meta*), les titres de rubriques (*h1* à *h6*) et normalisons simplement sans appliquer de transformation *tfidf*. Chaque catégoriseur est entraîné sur :

- (un maximum de) 300 exemples positifs choisis aléatoirement parmi tous les documents de la catégorie ;
- 10 000 exemples négatifs choisis aléatoirement sur toutes les autres catégories (30 exemples négatifs de chaque catégorie).

Dans les deux cas, nous entraînons un modèle de régression logistique (Yu et coll., 2011) à l'aide de l'outil LIBLINEAR (Fan et coll., 2008). Nous optons pour ce modèle régularisé, car il offre des performances proches des machines à vecteurs de support, tout en fournissant en sortie une probabilité d'appartenance au thème. Pour obtenir des performances optimales, nous ajustons deux paramètres du modèle par recherche quadrillée :

- le paramètre de régularisation C qui permet d'équilibrer sous/sur apprentissage ;
- le paramètre *W* qui permet d'éviter une sur pondération des classes majoritaires avec des jeux de données déséquilibrés.

L'évaluation du modèle et du choix des paramètres sont faits par double validation croisée stratifiée à 5 plis.

	Précision	Rappel	F_1-mesure
TextCat Toutes	0,87 (± 0,09)	0,48 (± 0,15)	0,61 (± 0,15)
WebPageCat Toutes	0,83 (± 0,10)	0,56 (± 0,15)	0,66 (± 0,14)
TextCat Top_{15}	0,95 (± 0,03)	0,68 (± 0,07)	0,79 (± 0,05)
WebPageCat Top_{15}	0,94 (± 0,03)	0,75 (± 0,07)	0,83 (± 0,05)

Tableau 6.1 : Macro-moyenne de la précision, du rappel et de la F_1-mesure sur le second niveau de l'OpenDirectory pour deux catégoriseurs s'appuyant sur le texte et la structure des pages Web. L'écart type correspondant à chaque moyenne est donné entre parenthèses. Les mesures sont fonctions du nombre de catégories étudiées (toutes ou Top_{15}).

Les résultats obtenus sur les 340 catégories de l'OpenDirectory montrent des performances variables en fonction des catégories. L'utilisation de la structure des pages Web permet un léger gain, entre 4 et 5 points de F_1-mesure (Tableau 6.1). Nous pouvons constater que ce gain est dû à une hausse du rappel (7 points). Néanmoins, étant donné que les deux classifieurs fonctionnent sur des traits différents (*tfidf* ou fréquences normalisés), il serait hasardeux de conclure sur les raisons de cette hausse.

Pour la suite de nos travaux, nous avons sélectionné quinze catégories ayant pour caractéristiques de traiter des domaines divers et d'offrir les meilleures performances de catégorisation : `Arts/Movies`, `Business/Accounting`, `Computers/Emulators`, `Games/Puzzles`, `Health/Dentistry`, `Home/Cooking`, `Recreation/Birding`, `Recreation/Camps`, `Science/Astronomy`, `Science/Chemistry`, `Shopping/Flowers`, `Shopping/Jewelry`, `Society/Genealogy`, `Sports/Golf`, `Sports/Martial_Arts`. Ces 15 catégories et les *crawls* annotés qui leurs sont associés seront utilisés dans la suite de nos travaux pour nos algorithmes d'apprentissage de fonctions d'ordonnancement. Les détails pratiques concernant la création de ces *crawls* est donné à la sous-section 6.2.2 (p. 105).

DES GRAPHES ANNOTÉS AUX LISTES DE DOCUMENTS ORDONNÉES

Nous nous intéressons à présent à la transformation des graphes de *crawl* annotés G_i en listes de liens ordonnés L_i. Chaque liste de liens doit refléter la frontière de *crawl* ordonnée par priorité de visite. Il semble tentant de générer plusieurs listes de liens correspondant aux différents états de la frontière *crawl* à différents moments du *crawl*. Toutefois, nous ne choisissons pas cette approche, car :

		Mène à une	
		Page pertinente	Page non pertinente
Est une	Page pertinente	3	2
	Page non pertinente	1	0

Tableau 6.2 : Échelle de pertinence utilisée pour notre évaluation.

– un *crawler* est distribué et ne peut donc être réduit à une unique entité parcourant le graphe ;
– la combinatoire générée par tous les états possibles est immense (elle dépend de l'itinéraire de chaque processus distribué et des amorces choisies).

Nous optons pour la production d'une liste partiellement ordonnée de documents à partir de l'intégralité de chaque graphe G_i. Dans cette liste, les liens directs aux documents pertinents pour un thème doivent systématiquement apparaître en tête de la frontière de *crawl* car ils offrent un gain maximal (un téléchargement égale un document pertinent). De plus, nous devons tenir compte du fait qu'un document puisse mener à d'autres documents pertinents : nous voulons favoriser le téléchargement d'un document non pertinent qui mènera à de nombreux documents pertinents par rapport à un document qui ne mènera à aucun document pertinent. Il serait pertinent d'ordonner la frontière encore plus finement en tenant compte de la distance à laquelle se trouvent des documents pertinents, ainsi que leur nombre. Cependant, cette approche demanderait d'avoir un graphe de liens relativement complet et des méthodes d'apprentissage adaptées à ces échelles de valeurs [13]. Nous limitons volontairement notre échelle à quatre valeurs qui sont choisies en fonction de la pertinence du lien (c'est à dire qu'il pointe vers une page pertinente) et l'apparition de pages pertinentes à une distance comprise entre un et trois liens. Cette échelle est résumée au Tableau 6.2.

EXTRACTION DE TRAITS

Dans un cadre de recherche d'information, les traits utilisés pour l'apprentissage de fonctions d'ordonnancement sont généralement classés en trois familles (Qin et coll., 2010) : *(i)* les traits dépendants d'une requête et d'un document ; *(ii)* les traits issus d'un document ; *(iii)* les traits issus d'une requête.

13. Les jeux de données en recherche d'information considèrent des échelles de pertinence discrètes et de petites tailles (Chapelle et Chang, 2011).

Nous repartons de cette classification, mais remplaçons les requêtes par des thèmes et les documents par des URL. Nous considérons donc trois types de traits : *(i)* les traits relatifs à un thème (notés T) ; *(ii)* les traits relatifs à une URL (notés U) ; *(iii)* les traits relatifs à un thème et une URL (notés T–U). De plus, lorsque nous évaluons la pertinence de télécharger une page (URL), nous considérons à la fois les traits spécifiques à l'URL (par exemple son texte d'ancrage), mais également les traits issus de la page où apparaît l'URL et de ses pages ancêtres.

Pour chaque URL, nous considérons les sources d'informations suivantes :
- URL de la destination
- URL de la page courante
- titre de la page : balise *title*
- corps de la page : balise *body*
- texte d'ancrage : balise *a*
- contexte du lien : fenêtre de 10, 20 et 40 mots autour du lien (Pant et Srinivasan, 2006)
- contenu informatif de la page : extrait par l'algorithme BodyTextExtraction (voir l'Annexe A)
- bloc de texte englobant le lien : première balise de type *block* [14] obtenue en remontant l'arbre DOM
- corps des pages parentes et ancêtres

La liste complète des traits extraits de ces sources est détaillée au Tableau 6.3. La position du lien dans le code HTML est calculée comme le ratio entre le numéro de ligne où apparaît le lien et le nombre de lignes total dans le fichier HTML. La distance en terme de répertoires est la différence entre le nombre de barres obliques de l'URL source et destination. La similarité sur les n-grammes de caractères est une similarité cosinus sur la fréquence des segments allant de 4 à 8 caractères (utilisés par Baykan et coll. (2009) pour la catégorisation thématique d'URL). La similarité sac de mots suit le modèle vectoriel classique avec similarité cosinus sur les mots pondérés par un *tfidf*. Le score de cohésion thématique est simplement la somme des poids des mots obtenus par la cohésion thématique. La détection de langue est calculée par un modèle Bayesien naïf sur les segments de 1 à 4 caractères (Lui et Baldwin, 2012). Enfin, notons que bien d'autres traits pourraient être utilisés tels que la position des liens dans la page après rendu visuel ou une similarité sémantique (Ramage et coll., 2009).

14. Voir la définition du W3C : http://www.w3.org/TR/html401/struct/global.html#didx-block-level

Tableau 6.3 : Traits extraits pour chaque entrée de la frontière de *crawl*.

	Type	Trait	Source
Traits relatifs à une URL			
1	U	Profondeur / Nombre de barres obliques	URL
2	U		URL page courante
3	U	Nombre de caractères	URL
4	U	Lien relatif ou absolu	URL
5	U	Lien interne ou externe	URL
6	U	Distance en terme de répertoires (barres obliques)	URL
7	U	Position du lien dans le code HTML	URL
8	U	Position normalisée du lien dans l'arbre DOM	URL
9	U	Lien dans la zone de contenu informatif	URL
10	U	Nombre de mots	URL
11	U		URL page courante
12	U		Texte d'ancrage
13-15	U		Contexte du lien
16	U		Titre de la page
17	U		Corps de la page
18	U		Contenu informatif
19	U		Bloc englobant
20	U	Nombre de liens	Page courante
21	U		Bloc englobant
22	U	Image comme texte d'ancrage [15]	Texte d'ancrage
23-24	U	Taille de l'image (si renseignée dans le code HTML) [15]	Texte d'ancrage
25	U	Probabilité que la page soit en anglais	Corps de la page
Traits relatifs à un thème et une URL			
26	T–U	Similarité sur les n-grammes de caractères	URL
27	T–U		URL page courante
28	T–U	Similarité sac de mots	URL
29	T–U		URL page courante
30	T–U		Texte d'ancrage
31-33	T–U		Contexte du lien
34	T–U		Titre de la page
35	T–U		Corps de la page
36	T–U		Contenu informatif
37	T–U		Bloc englobant
38	T–U	Somme des poids de la cohésion thématique	URL
39	T–U		URL page courante
40	T–U		Texte d'ancrage

15. D'après SEO by the sea.

Tableau 6.3 : Traits extraits pour chaque entrée de la frontière de *crawl*.

	Type	Trait	Source
41-43	T–U		Contexte du lien
44	T–U		Titre de la page
45	T–U		Corps de la page
46	T–U		Contenu informatif
47	T–U		Bloc englobant
48	T–U	Régression logistique	Page courante
49	T–U		Page parente
50	T–U		Page grand-parente

APPRENTISSAGE DE FONCTIONS D'ORDONNANCEMENT

Les méthodes d'apprentissage de fonctions d'ordonnancement peuvent être divisées en trois familles (Li, 2011, chap. 2, p. 21) :

- Approches *pointwise* : apprendre à prédire la pertinence d'entrées ;
- Approches *pairwise* : apprendre à ordonner des paires d'entrées ;
- Approches *listwise* : apprendre à ordonner la liste d'entrées.

Il est admis que les approches *pairwise* et *listwise* surpassent les méthodes *pointwise* (Li, 2011, chap. 2, p. 22). Toutefois, cette assertion dépend des modèles sous-jacents utilisés. Ainsi, une approche *pointwise* fondée sur un modèle non linéaire état de l'art peut surpasser une approche *pairwise* utilisant un modèle linéaire (Chapelle et Chang, 2011, Tableau 5).

Nous proposons d'évaluer quatre méthodes, réparties sur l'ensemble des trois familles de méthodes :

1. GBRT (Friedman, 2001), approche *pointwise* fondée sur une régression non linéaire ;
2. RankSVM (Herbrich et coll., 1999), approche *pairwise* fondée sur une classification linéaire ;
3. Coordinate Ascent[16] (Metzler et Croft, 2007), approche *listwise* linéaire ;
4. LambdaMART (Wu et coll., 2010), approche *listwise* non linéaire.

Les arbres de décision boostés (*Gradient Boosted Regression Trees* ou GBRT) sont un algorithme de *boosting* utilisant des arbres de décision comme

16. Nous utilisons le terme « *Coordinate Ascent* » pour désigner cette méthode bien que ce terme réfère à une technique d'optimisation qui n'est pas spécifique à l'apprentissage de fonctions d'ordonnancement.

classifieur faible. Ce modèle n'est pas spécifique à l'apprentissage de fonctions d'ordonnancement et ignore la distinction entre thèmes pour faire de la régression sur les scores de pertinence. Il nécessite la calibration de plusieurs paramètres permettant d'équilibrer sur/sous apprentissage : le nombre d'arbres, la taille des arbres, le taux d'apprentissage et le taux d'échantillonnage. Nous invitons le lecteur intéressé à se référer à l'article de Ganjisaffar et coll. (2011, section 2) pour une description concise et claire de ces paramètres.

L'algorithme RankSVM (parfois nommé Ranking SVM ou SVMRank) formalise le problème d'apprentissage comme un problème de classification binaire entre paires de documents et s'appuie sur des machines à vecteurs de supports (SVM) pour le résoudre. Cette approche a obtenu des résultats intéressants dans le cadre de la campagne d'évaluation LETOR (Qin et coll., 2010).

Coordinate Ascent (CA) utilise une méthode d'optimisation itérative pour trouver les paramètres d'un modèle linéaire. À chaque itération, tous les paramètres sauf un sont fixés et le dernier est optimisé en fonction des performances de la méthode sur la liste complète d'entrées. Pour éviter les minima locaux, l'optimisation est lancée à plusieurs reprises en partant de points de départ différents. C'est un algorithme simple et performant pour les petits ensembles de traits. De plus, aucune phase de validation n'est nécessaire.

Enfin, LambdaMART est également une méthode fondée sur des arbres de décision boostés. Cependant, les paramètres des arbres sont ajustés en fonction des performances sur la liste d'entrées complète. Cette méthode a obtenu les meilleurs résultats dans le cadre de la campagne d'évaluation *Yahoo ! learning-to-rank challenge* (Chapelle et Chang, 2011).

6.2.2 *Évaluations*

Nous évaluons l'apport de notre méthode au travers de deux expériences. Une première expérience qui a pour objectif de comparer les différents modèles d'apprentissage de fonctions d'ordonnancement ainsi que leurs paramètres optimaux. Une seconde expérience en conditions réelles de collecte sur le Web pour différents thèmes de l'OpenDirectory.

DONNÉES

Comme nous l'avons mentionné précédemment, nous avons collecté un ensemble de documents pour chacune des 15 catégories sélectionnées.

Nous avons démarré chaque *crawl* à partir de 20 URL amorces sélectionnées aléatoirement à partir de l'OpenDirectory. Le *crawler* itère ensuite des phases de téléchargement par lots (300 nouvelles URL) et d'analyse des pages téléchargées. L'analyse des pages a pour objectif d'extraire de nouveaux liens que nous pondérons *via* l'algorithme Shark-Search. Nous sélectionnons alors 10 liens sortants aléatoirement tout en respectant la distribution des poids. Nous pensons que cette approche permet d'obtenir un échantillon représentatif des liens des pages et que les 10 liens fournissent une estimation de la propension de cette page à mener à des pages pertinentes. La procédure de collecte est arrêtée lorsque 10 000 documents ont été téléchargés.

Durant le *crawl*, les pages téléchargées sont nettoyées à l'aide de l'outil lxml [17] (suppression des balises de style, de scripts, de lien (`<link>`), de formulaire, de commentaires) et analysées *via* la bibliothèque BeautifulSoup [18].

La collecte achevée, nous disposons, pour chaque thème, d'un ensemble de 10 000 pages Web analysées et du graphe de liens hypertextes parcouru par le *crawler*. Une étape d'analyse supplémentaire nous permet de compléter ce graphe d'hyperliens en considérant tous les hyperliens des pages Web. Les graphes d'hyperliens complets contiennent alors environ 300 000 arcs en moyenne sur nos 15 catégories. Le degré sortant moyen sur l'ensemble de nos *crawls* est 189 (± 77), bien loin des estimations de Kumar et coll. en 2000 qui observaient un degré sortant moyen à $7,2$. Notons qu'après 5 000 pages téléchargées, la frontière de *crawl* contient déjà près d'un million d'URL, ce qui montre bien l'importance de la priorisation de la frontière de *crawl*.

Nous appliquons ensuite l'étape d'extraction et de normalisation des traits. Les traits relatifs à la taille des images dans les liens sont absents de 97% des entrées. Nous les utilisons comme tels (*N/A*) avec les arbres de décision boostés (GBRT), mais les supprimons pour les autres algorithmes qui utilisent un format de fichier ne permettant pas de spécifier les valeurs manquantes. Les quelques autres valeurs manquantes (par exemple la similarité avec une balise manquante) sont remplacées par des valeurs nulles. Nous normalisons ensuite les traits dont la valeur n'est pas comprise entre 0 et 1 (cohésion thématique, nombre de mots ou de liens, etc.), car cela pourrait affecter les performances de certains de nos modèles. Enfin, nous divisons nos 15 thèmes en 5 groupes de 3 thèmes. Trois groupes sont utilisés pour l'entraînement (9 thèmes) et les deux restants pour l'éva-

17. http://lxml.de/
18. http://www.crummy.com/software/BeautifulSoup/

	Requêtes (Thèmes)	Documents (Liens)	Niveaux de pertinence	Traits
Letor 3.0 – Ohsumed	106	16 k	2	45
Letor 3.0 – Gov	575	568 k	2	64
Letor 4.0	2 476	85 k	3	46
Yandex	20 267	213 k	5	245
Yahoo !	36 251	883 k	5	700
Microsoft	31 531	3,8 M	5	136
Notre corpus	15	5,4 M	4	50

Tableau 6.4 : Comparaison de notre jeu de données avec plusieurs jeux de données publiques d'apprentissage de fonctions d'ordonnancement.

luation (3 thèmes) et la calibration des paramètres (3 thèmes). Cinq plis différents sont générés par roulement. Les mesures d'évaluation rapportées ci-après sont les moyennes des mesures obtenues sur chacun des plis.

Le Tableau 6.4 (d'après Chapelle et Chang, 2011) présente les caractéristiques principales de notre jeu de données comparativement à plusieurs jeux de données publiques de recherche d'information. Contrairement à ces jeux de données, nous constatons que notre corpus possède un nombre de requêtes (thèmes) faible, mais un grand nombre de documents (liens). Le nombre de liens très élevé dans notre corpus nous impose d'échantillonner nos données pour entraîner nos algorithmes. Aslam et coll. (2009) ont montré que plusieurs algorithmes d'apprentissage de fonctions d'ordonnancement entraînés sur une sélection aléatoire des entrées du corpus LETOR (stratégie dénommée *infAP* dans leur article) obtenaient une précision moyenne équivalente aux mêmes algorithmes entraînés sur l'ensemble des données.

Nous échantillonnons 500 000 entrées aléatoirement (env. 15 % du corpus d'entraînement) pour entraîner les algorithmes GBRT, CA et LambdaMART. Ce nombre a été choisi pour être minimal sans montrer pour autant de baisse de performances. Nous avons utilisé l'outil scikit-learn [19] pour le modèle GBRT et RankLib [20] pour les modèles CA et LambdaMART.

Pour l'algorithme RankSVM, qui nécessite la construction de paires d'entrées, nous choisissons une approche stochastique qui ne construit qu'un petit ensemble de paires à partir du corpus initial parmi toutes les combinaisons possibles (Sculley, 2009). En pratique, nous utilisons l'outil sofia-ml [21] de Sculley. L'auteur utilisait dans son article 100 000 itérations pour

19. http://scikit-learn.org/
20. http://people.cs.umass.edu/~vdang/ranklib.html
21. https://code.google.com/p/sofia-ml/

entraîner son classifieur sur le corpus LETOR 4.0. Nos premiers tests en validation croisée ont montré qu'un échantillon d'un million de paires suffisait à obtenir des performances optimales sur notre corpus. Nous utilisons donc cette valeur dans la suite de nos expériences. Par ailleurs, l'échantillonnage des paires est obtenu *via* un index qui permet d'obtenir efficacement l'ensemble des exemples correspondant à un thème et une valeur de pertinence.

MESURES D'ÉVALUATION

Pour comparer les performances de différents algorithmes d'ordonnancement, nous nous tournons vers trois mesures complémentaires permettant d'évaluer des listes ordonnées : la précision moyenne (AP/MAP), le gain cumulé normalisé (NDCG) et le τ de Kendall. L'ordonnancement des entrées étant évalué indépendamment pour chaque thème, nous calculons une macro moyenne de ces mesures sur l'ensemble des thèmes.

La précision moyenne, que nous avons déjà utilisée au Chapitre 4 (§ 4.3, p. 57), fonctionne sur une échelle de pertinence binaire. Nous l'appliquerons donc en ramenant nos valeurs de pertinence au cas binaire (page pertinente / non pertinente).

Le gain cumulé normalisé (Järvelin et Kekäläinen, 2002), au contraire, a été défini pour évaluer des échelles de pertinence non binaires. Formellement, il est défini comme :

$$\mathrm{NDCG} = \frac{\mathrm{DCG}}{\mathrm{IDCG}} \qquad \mathrm{DCG} = \sum_i \frac{2^{\mathrm{rel}_i} - 1}{\log_2(i+1)}$$

avec IDCG, le gain cumulé de l'ordre optimal, et rel_i la valeur de pertinence prédite pour le document i, comprise entre 0 et 3 dans notre cas. Cette mesure attribue une importance exponentiellement haute aux documents pertinents. Elle sera donc intéressante pour évaluer la capacité de notre méthode à placer les documents les plus pertinents en tête de la frontière de *crawl*.

Enfin, le τ de Kendall (Knight, 1966) est une mesure issue des statistiques qui s'appuie essentiellement sur le nombre de paires inversées pour évaluer la corrélation entre deux ordonnancements. Elle est en ce sens similaire au ρ de Spearman que nous avons utilisé précédemment, mais est plus largement utilisée par la communauté d'apprentissage de fonctions d'ordonnancement. Nous utilisons la mesure τ_b qui inclut une correction pour les paires de rangs égaux.

$$\tau_b = \frac{(P - Q)}{\sqrt{(P + Q + T_1) * (P + Q + T_2)}}$$

avec P le nombre de paires concordantes, Q le nombre de paires en désaccord, T_1 le nombre de rangs égaux dans la première liste et T_2 le nombre de rangs égaux dans la seconde liste. Contrairement à la NDCG, cette mesure ne tient pas compte de la valeur de pertinence. Elle nous sera utile pour évaluer plus globalement la qualité de l'ordre.

APPRENTISSAGE

Pour chaque méthode, nous avons sélectionné, pour chaque pli, les paramètres offrant le NDCG optimal sur l'ensemble de validation. Nous présentons au Tableau 6.5 l'importance relative (en pourcentage) de chaque trait fournie par l'algorithme GBRT. Nous avons choisi de présenter les traits de cette méthode, car :

- La sélection d'attributs est inhérente à l'algorithme ;
- La méthode de pondération est facilement interprétable ;
- L'information était facilement accessible depuis l'outil.

Dans les arbres de décision boostés, l'importance relative est dérivée de la profondeur moyenne des traits dans l'ensemble des arbres de décision construits (Friedman, 2001). Les résultats que nous présentons sont les moyennes des valeurs obtenues sur les 5 plis. Nous observons des traits très discriminants envers les pages pertinentes en haut du classement (page courante en langue non anglaise, image dans le texte d'ancrage). Nous constatons également un poids important accordé à la pertinence de la page grand-parente. Nous supposons que cette information est importante pour classer les pages ayant une pertinence de 1 (tunnels). Enfin, ces résultats semblent valider une nouvelle fois l'importance de prendre en compte différents niveaux d'informations pour ordonner la frontière de *crawl*.

VALIDATION CROISÉE

Nous évaluons à présent nos quatre algorithmes sur les 5 plis créés précédemment (Tableau 6.6). Nous constatons un NDCG très haut pour tous les algorithmes. Le calcul de cette mesure étant dominé par la validité de la tête de liste, ces valeurs montrent une bonne capacité à placer des documents pertinents en tête de liste. Ce résultat est d'ailleurs confirmé par la MAP, également haute qui montre que les documents pertinents se trouvent bien en tête de liste. Les valeurs positives et moyennes du τ de Kendall, en complément d'une MAP et d'un NDCG très hauts nous montrent une difficulté plus importante à classer les documents dans la seconde partie de la liste. Pour confirmer ce résultat et évaluer la capacité des algorithmes à favoriser les liens tunnel, nous évaluons la MAP uniquement sur les liens non pertinents, c'est à dire ayant un score de pertinence de 0 ou de 1 (Tableau 6.2). Sur les données de test, nous obtenons des

Rang	Trait	Poids (%)
1	Probabilité que la page soit en anglais	6,7%
2	Régression logistique sur la page grand-parente	6,7%
3	Image comme texte d'ancrage	6,1%
4	Similarité sac de mots sur un contexte de 20 mots	4,7%
5	Lien dans la zone de contenu informatif	4,1%
6	Similarité n-grammes sur l'URL	3,8%
7	Longueur du contexte de 10 mots [22]	3,5%
8	Cohésion thématique du le bloc englobant	3,3%
9	Longueur du titre de la page courante	3,3%
10	Similarité sac de mots sur un contexte de 10 mots	3,1%
11	Régression logistique sur la page courante	2,7%
12	Similarité sac de mots sur l'URL cible	2,6%
13	Similarité sac de mots sur l'URL courante	2,0%
14	Longueur du contexte de 20 mots [22]	2,0%
15	Lien interne au site	1,9%

Tableau 6.5 : Importance relative des 15 meilleurs traits pour l'algorithme GBRT.

valeurs de $0,77$ et $0,81$, plus faibles que les précédentes valeurs de MAP. Cependant, ces valeurs toujours hautes montrent une bonne capacité à favoriser les tunnels, et par là même, montrent l'intérêt de notre approche.

Ces résultats montrent aussi la supériorité des arbres de décision boostés sur les modèles linéaires. En effet, sur les données de test, l'approche *pointwise* GBRT surpasse l'approche *pairwise* RankSVM, ce qui avait déjà été constaté par Chapelle et Chang (2011). De même l'approche LambdaMART également basée sur des arbres de décision boostés surpasse l'approche CA. Nous pensons que cette différence de performances est due à la capacité des arbres de décision à prendre en compte les intéractions entre les traits (modèle non linéaire). Cette hypothèse est soutenue par le fait que nous avons obtenu de meilleures performances en considérant des arbres relativement profonds (hauteur entre 5 et 10).

Ces résultats démontrent également l'intérêt des approches *listwise*, qui optimisent directement la mesure NDCG sur l'intégralité de la liste. Ces modèles obtiennent ici les meilleures performances avec un gain de 9 points sur la précision moyenne et 12 points sur le τ de Kendall par rapport à l'algorithme *pairwise* RankSVM. Dans la suite de nos travaux, nous nous limiterons à l'algorithme LambdaMART, qui a obtenu les meilleurs résultats.

22. La taille des contextes est fixe, mais peut varier sur certaines pages. C'est notamment le cas sur des pages vides de texte par exemple.

	Validation			Test			
	MAP	τ	NDCG	MAP	τ	NDCG	$MAP_{\{0,1\}}$
RankSVM	**0,89**	0,59	0,99	0,80	0,52	0,98	0,77
GBRT	0,85	0,56	0,99	0,82	0,56	0,98	0,81
Coordinate Ascent	0,88	0,61	0,99	0,88	0,60	0,99	0,77
LambdaMART	0,88	**0,63**	0,99	**0,89**	**0,64**	0,99	0,81

Tableau 6.6 : Performances des quatre algorithmes d'apprentissage de fonctions d'ordonnancement sur les ensembles de test et de validation après ajustement des paramètres.

ÉVALUATION EN CONDITIONS RÉELLES

Fort de nos premières conclusions quant aux algorithmes d'apprentissage de fonctions d'ordonnancement, nous comparons à présent notre approche avec plusieurs stratégies de *crawl* :

- Un parcours en largeur ;
- Une stratégie fondée sur l'analyse de liens (OPIC) ;
- Une stratégie de seconde génération utilisant à la fois la pertinence de la page parente et le contexte des liens (Shark-Search).

Le fonctionnement de la stratégie Shark-Search peut être ajusté au travers de quatre paramètres : d, b, g et la taille du contexte des liens (Hersovici et coll., 1998, fig. 2). Dans nos expériences, nous utilisons les valeurs $d = 0,5$ et $b = 0,8$ à l'instar de Hersovici et coll. Comme Menczer et coll. (2003), nous fixons g à $0,1$ selon les conclusions de Hersovici et coll.. Enfin, nous utilisons arbitrairement un contexte de 10 mots.

Notre objectif est d'évaluer le nombre de pages pertinentes téléchargées en fonction du temps. Nous avons considéré plusieurs options pour évaluer la pertinence des pages :

- Appliquer la méthodologie de Pant et Srinivasan (2006) ou Srinivasan et coll. (2005) qui s'appuient sur l'apparition des documents de l'OpenDirectory durant le *crawl* ;
- Appliquer un catégoriseur thématique entraîné sur l'OpenDirectory (Chakrabarti et coll., 2002).

Nos expériences préliminaires avec la première solution ont montré que nous ne rencontrions que trop peu de pages de l'OpenDirectory durant nos *crawl*. Par conséquent, nous jugeons les mesures obtenues peu fiables et optons pour la seconde solution. Bien que l'utilisation d'un catégoriseur automatique implique une référence imparfaite, notre objectif est de

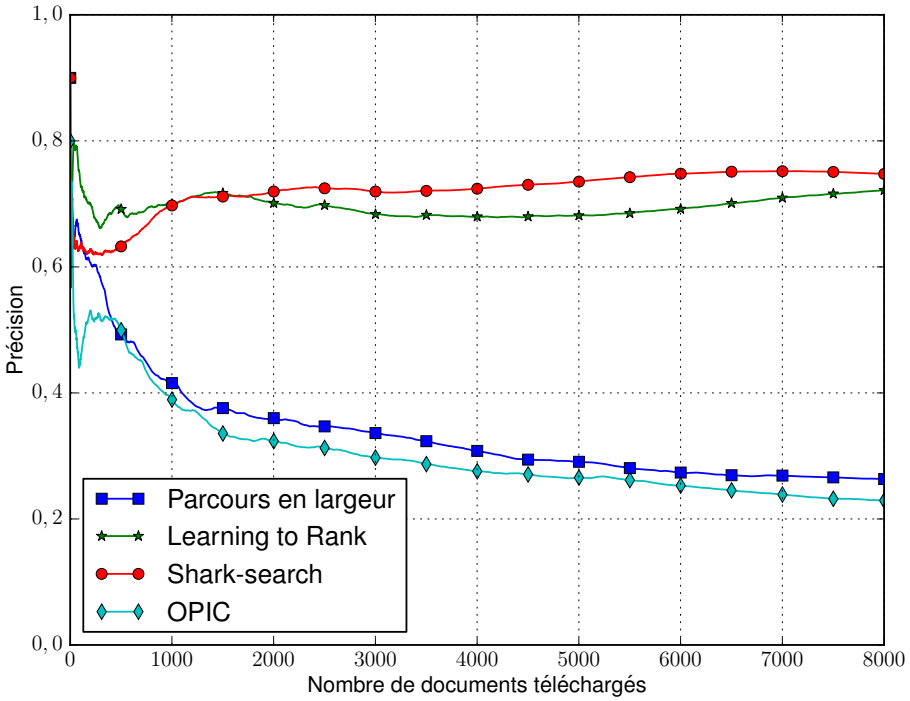


FIGURE 6.5 : Précision des quatre *crawlers* en fonction du nombre de documents téléchargés.

comparer les performances de différents *crawlers* entre eux. Nous pensons donc que le biais dû aux performances des catégoriseurs est négligeable dans notre contexte.

Nous comparons nos *crawlers* sur dix nouvelles catégories, toujours extraites du second niveau de l'OpenDirectory : `Arts/Bodyart`, `Business/Aerospace_and_Defense`, `Games/Card_Games`, `Health/Pharmacy`, `Home/Gardening`, `Recreation/Boating`, `Shopping/Tools`, `Society/Death`, `Society/Law`, `Sports/Fencing`. Comme précédemment, nous avons choisi ces catégories en fonction de leur répartition et des performances raisonnables des catégoriseurs sur ces catégories.

Les *crawlers* sont démarrés à partir des mêmes 20 URL amorces et téléchargent 300 nouvelles URL à chaque itération jusqu'à obtenir un total de 8 000 documents. Une fois téléchargés, les 8 000 documents sont ordonnés en fonction de leur date de téléchargement et la mesure de précision à N est calculée pour chaque nouvelle page téléchargée. Une moyenne des mesures est ensuite calculée sur l'ensemble des catégories. Les performances

sont présentées à la Figure 6.5.

Nous constatons que le parcours en largeur et le critère OPIC, qui n'intègrent aucune information thématique, obtiennent très rapidement une précision inférieure à 0,5. Au contraire, les stratégies orientées obtiennent une précision sensiblement plus haute, aux alentours de 0,7. Notre approche obtient la meilleure précision sur les 1 500 premiers documents. Elle est ensuite surpassée par l'approche Shark-Search. Bien que leur précision soit comparable, ce résultat est étonnant, car notre approche inclut notamment les traits utilisés par la stratégie Shark-Search. Nous formulons plusieurs hypothèses quant aux raisons menant à ces résultats :

- Certains des traits que nous avons choisis introduisent du bruit (traits non pertinents, normalisation des valeurs).
- La durée des *crawls* est trop courte pour évaluer l'apport d'une bonne prise en charge de l'effet tunnel.

De plus, de par la variabilité des résultats obtenus sur les différentes catégories, nous pensons qu'une étape de validation sur un plus grand nombre de thèmes s'avère nécessaire pour statuer définitivement sur les performances de ces approches.

6.3 Bilan

Dans ce chapitre, nous nous sommes tournés vers une approche de collecte alternative : l'exploration orientée. Nous avons tout d'abord présenté notre *crawler* orienté avant de nous attarder sur différents aspects de ce dernier (caractère distribué, robustesse). Nous avons ensuite proposé d'améliorer la stratégie d'ordonnancement de la frontière de *crawl* qui est au cœur des *crawler* orientés. Nous avons défini une méthode permettant d'exploiter des *crawls* existants pour entraîner une fonction d'ordonnancement efficace. Cette fonction d'ordonnancement est apprise *via* un algorithme d'apprentissage automatique qui nous permet de tirer parti de traits divers et de modéliser l'effet tunnel. Nos expériences en validation croisée ont montré une bonne capacité à ordonner les liens pertinents, tunnels ou non pertinents. Cependant, l'application de notre méthode pour la collecte de dix nouveaux thèmes a montré des performances proches mais inférieures à une stratégie état de l'art Shark-Search. Nous avons formulé plusieurs hypothèses quant à ces résultats qui devront être explorées pour statuer définitivement sur la pertinence de notre méthode.

Nous espérons que nos travaux sur l'apprentissage de fonctions d'ordonnancement pour le *crawling* orienté pavent la voie vers une nouvelle génération de *crawlers* permettant de tirer parti de nombreuses sources

d'information pour orienter leurs parcours. Ces travaux ouvrent en tout cas de nombreuses perspectives :

1. Tout d'abord, nous avons fait l'hypothèse que nous pouvions définir un modèle d'ordonnancement unique, indépendant du thème. Nous pensons que cette approche est simplificatrice, car notre modèle ne peut apprendre les liens entre thèmes, une information qui semble pertinente pour la prise en compte de l'effet tunnel (Chakrabarti et coll., 2002; Diligenti et coll., 2000).

2. Nous avons fait le choix d'apprendre notre fonction d'ordonnancement à partir de corpus annotés réalisés spécifiquement pour notre expérience. Cependant, les graphes d'hyperliens de ces *crawls* sont incomplets. Pour estimer la propension des liens à être des tunnels de manière plus précise, il serait plus intéressant de nous appuyer sur des graphes d'hyperliens plus complets. Cette approche nécessiterait cependant de traiter un très grand nombre de pages (plusieurs millions) et donc de liens entre les pages (milliards). La complexité technique et l'adéquation des algorithmes et outils d'apprentissage de fonctions d'ordonnancement actuels avec le nombre de données à traiter sera un frein important à cette piste.

3. Nous avons considéré que nos catégoriseurs thématiques permettaient de générer une référence avec une précision suffisante. Il serait intéressant d'entraîner et d'évaluer notre approche sur des graphes de *crawls* annotés manuellement. Les annotations manuelles réalisées dans le cadre du projet TTC sur les thèmes de l'énergie éolienne et des téléphones mobiles devraient nous permettre d'évaluer le biais introduit par nos catégoriseurs.

4. Une autre limitation de notre approche est son incapacité à intégrer des traits dynamiques qui sont fonction de la progression du *crawl*. Cette famille de traits inclut par exemple les traits liés à l'analyse du graphe de liens (degré entrant, PageRank) ou la co-citation (si une page téléchargée s'avère pertinente, favoriser ses pages sœurs). Alors que nous considérons l'ensemble du graphe de *crawl* comme corpus d'apprentissage, il serait certainement plus intéressant d'extraire des traits à différents moments de ce *crawl*, ce qui nous permettrait d'intégrer les traits dynamiques. Dans le même ordre d'idées, une même URL rencontrée plusieurs fois sur une même page ou dans un ensemble de pages constitue autant d'entrées différentes. Il serait en effet hasardeux de calculer une moyenne, ou un maximum sur les traits extraits pour cette URL à différents endroits. Agréger

l'ensemble d'informations que nous avons sur chaque URL demeure donc une piste ouverte.

CONCLUSION

Synthèse

Les moteurs de recherche verticaux, qui se concentrent sur des segments spécifiques du Web, deviennent aujourd'hui de plus en plus présents dans le paysage d'Internet. Les moteurs de recherche thématiques, notamment, visent à améliorer la pertinence des résultats, à réaliser des recherches plus fines et à diminuer le temps de recherche en limitant leur index à un segment thématique du Web. Dans ce manuscrit, nous nous sommes intéressé à la création de moteurs de recherche thématiques, en nous concentrant sur la collecte de documents à partir du Web. Après avoir étudié l'existant, nous avons conclu que deux méthodes majeures pour la collecte de documents thématiques étaient la recherche orientée, qui formule des requêtes à un moteur de recherche existant pour accéder aux documents du Web, et l'exploration orientée, qui consiste à parcourir le Web d'hyperlien en hyperlien à la recherche de documents pertinents.

Nous avons tout d'abord concentré nos efforts sur la recherche orientée. Nous avons défini un cadre d'évaluation objectif et reproductible en nous appuyant sur un grand nombre de thèmes issus de l'OpenDirectory. Puis, nous avons utilisé ce cadre pour évaluer les performances d'une approche standard pour la collecte de documents thématiques *via* un moteur de recherche : l'approche par combinaisons de termes. Nous avons constaté des performances variables en fonction de la taille des requêtes et une F_1-mesure optimale pour des requêtes composées de trois termes. En outre, nous n'avons observé aucune tendance claire quant à la difficulté de groupes thématiques de catégories.

Afin d'améliorer ces premiers résultats, nous avons identifié deux points charnières : en amont du moteur de recherche et en aval du moteur de recherche. En amont du moteur de recherche, notre objectif est d'améliorer la qualité des requêtes formulées et plus particulièrement des termes utilisés pour former les requêtes. Nous avons donc en premier lieu défini un critère de cohésion thématique original permettant de sélectionner des termes amorces plus discriminants. Ce critère est calculé en deux temps : un ensemble de documents contenant chaque terme est collecté à partir d'un moteur de recherche. Puis, nous modélisons les cooccurrences de ces termes sous la forme d'un graphe orienté et appliquons un algorithme de marche aléatoire afin d'assigner à chaque terme une mesure globale

de son importance pour le thème. Nous avons évalué l'influence de différents paramètres tels que le nombre de documents, l'utilisation de *snippets* ou le nombre de termes, sur la mesure proposée. Nous avons ensuite appliqué notre critère pour la sélection de termes amorces et de requêtes. Nous avons observé une corrélation forte entre notre critère et la précision moyenne des requêtes composées d'un unique terme. Nous avons ensuite évalué l'apport de notre critère de cohésion thématique dans le cadre d'évaluation que nous avons préalablement défini et avons observé un gain en précision ainsi qu'une baisse du rappel.

En aval du moteur de recherche, nous avons proposé d'appliquer un filtre aux documents téléchargés afin d'améliorer la qualité des corpus produits. Nous avons alors défini une procédure unifiée (GrawlTCQ) fondée sur un modèle de graphe hétérogène triparti qui permet à la fois d'ordonner les documents par pertinence et de sélectionner de nouveaux termes amorces pour étendre l'ensemble de documents collecté. Nous avons démontré l'intérêt de notre méthode par le biais de deux expériences reposant sur deux jeux de données distincts : l'OpenDirectory et un corpus de dépêches d'actualités de l'Agence France Presse. Nos résultats montrent que GrawlTCQ peut permettre d'améliorer la qualité des corpus produits en ordonnant et en ne conservant que les documents les mieux classés. De plus, GrawlTCQ permet d'améliorer de manière notable la sélection de termes amorces pour la recherche orientée itérative.

Pour pallier les limites liées à l'utilisation d'un moteur de recherche Web, nous nous sommes ensuite tournés vers l'exploration orientée du Web. Nous avons dans un premier temps présenté Babouk, notre *crawler* orienté, qui nous sert également de plateforme de test pour nos expériences. Nous avons ensuite proposé d'apprendre une fonction d'ordonnancement pour prioriser la frontière de *crawl*. Nous avons décrit une méthode permettant de tirer parti de données de *crawls* existants pour réaliser cet apprentissage. Nous avons ensuite sélectionné cinquante traits et comparé les performances de plusieurs algorithmes d'apprentissage en validation croisée avant d'appliquer la fonction d'ordonnancement apprise sur plusieurs *crawls* en conditions réelles. Nos résultats montrent des performances intéressantes, proches d'un algorithme de *crawling* orienté état de l'art sans pour autant le surpasser sur des *crawls* de 10 000 documents.

Perspectives

Nous allons présenter quelques unes des perspectives que nous envisageons pour poursuivre ce travail sur la collecte orientée pour la création

de moteurs de recherche thématiques.

Nous nous sommes dirigés, dès le début de nos travaux sur la recherche orientée, vers des méthodes de collecte automatique ne nécessitant que peu de données en entrée. Cependant, il semble légitime d'envisager un second cas de figure où un ensemble de données plus important serait fourni en entrée du système. Nous pensons par exemple à quelques exemples de documents Web (collectés manuellement ou issus de l'OpenDirectory) qui serviraient à entraîner un premier catégoriseur thématique. Ce type d'approche nous permettrait entre autres d'appliquer un filtrage plus strict en aval des moteurs de recherche sur les corpus téléchargés.

Notre première expérience sur l'apprentissage de fonctions d'ordonnancement pour l'exploration orientée a montré des résultats inférieurs à l'état de l'art. Nous pensons que cette expérience doit être reproduite dans un cadre différent pour montrer l'apport de notre prise en compte de l'effet tunnel. De plus, cette expérience devrait être reproduite sur un plus grand nombre de thèmes, comme nous l'avons fait pour nos expériences de recherche orientée. Par ailleurs, cette dernière réflexion nous amène naturellement au constat que nous avons évalué la recherche orientée et l'exploration orientée sur les même données mais de manières différentes. Ces deux approches ont pourtant un même but et il serait dès lors intéressant de les évaluer comparativement afin de mieux observer les limites de chaque approche.

Dans cette thèse, nous n'avons pas abordé les réseaux sociaux qui sont en même temps une nouvelle source d'information et un véritable vecteur de popularité pour les pages et sites Web. À l'époque actuelle, il semble primordial de faire usage de ces informations, que ce soit dans les algorithmes de classement des pages ou directement dans l'interface utilisateur du moteur de recherche thématique en valorisant les utilisateurs experts du domaine.

Enfin, nos recherches se sont concentrées sur la collecte de documents spécialisés, phase nécessaire, complexe et passionnante préalable à la création de moteurs de recherche thématiques. De tels moteurs peuvent et doivent ensuite tirer parti de la spécialisation des données qu'ils indexent pour s'imposer faces aux moteurs de recherche généralistes. Pour ce faire, nous considérons les éléments suivants comme primordiaux : *(i)* proposer un outil de recherche adapté au domaine étudié ; *(ii)* disposer d'une fonction de classement spécifique au domaine étudié ; *(iii)* intégrer des connaissances du domaine étudié. Ces trois pistes rejoignent des domaines de re-

cherche variés et ouvrent un grand nombre de perspectives qui restent à explorer.

BIBLIOGRAPHIE

Abiteboul, S., Preda, M. et Cobena, G. (2003). Adaptive on-line page importance computation. *Dans Proceedings of the 12th international conference on the World Wide Web*, pages 280–290, New York, New York, USA. ACM Press. Cité pages 23 et 92.

Agichtein, E. et Gravano, L. (2000). Snowball : Extracting relations from large plain-text collections. *Dans Proceedings of the 5th ACM conference on Digital libraries*, pages 85–94. ACM. Cité page 18.

Agirre, E. et Soroa, A. (2009). Personalizing PageRank for word sense disambiguation. *Dans Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 33–41. Cité page 74.

Ahmad, K., Gillam, L. et Tostevin, L. (1999). University of surrey participation in trec 8 : Weirdness indexing for logical document extrapolation and retrieval (wilter). *Dans The Eighth Text REtrieval Conference (TREC-8)*. Cité page 84.

Alonso, A., Blancafort, H., de Groc, C., Million, C. et Williams, G. (2012). Metricc : Harnessing comparable corpora for multilingual lexicon development. *Dans Proceedings of the 15th EURALEX International Congress*, pages 389–403. Cité page 89.

Arguello, J., Diaz, F. et Callan, J. (2011). Learning to Aggregate Vertical Results into Web Search Results. *Dans Conference on Information and Knowledge Management*. Cité page 12.

Arguello, J., Diaz, F., Callan, J. et Crespo, J. (2009). Sources of evidence for vertical selection. *Dans Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 315–322. ACM. Cité page 12.

Aslam, J., Kanoulas, E., Pavlu, V., Savev, S. et Yilmaz, E. (2009). Document selection methodologies for efficient and effective learning-to-rank. *Dans Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 468–475. ACM. Cité page 107.

Babaria, R., Nath, J., Bhattacharyya, C., Murty, M. et coll. (2007). Focused crawling with scalable ordinal regression solvers. *Dans Proceedings*

of the 24th international conference on Machine learning, pages 57–64. ACM. Cité page 26.

Baeza-Yates, R., Castillo, C., Marin, M. et Rodriguez, A. (2005). Crawling a country : better strategies than breadth-first for web page ordering. *Dans Special interest tracks of the 14th international conference on World Wide Web*, pages 864–872. ACM. Cité pages 22 et 92.

Baeza-Yates, R. et Ribeiro-Neto, B. (1999). *Modern information retrieval*, volume 463. ACM press New York. Cité page 10.

Banko, M. et Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. *Dans Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 26–33. Association for Computational Linguistics. Cité page 90.

Bar-Yossef, Z. et Gurevich, M. (2006). Random sampling from a search engine's index. *Dans Proceedings of the 15th international conference on the World Wide Web*, page 367, New York, New York, USA. ACM Press. Cité page 94.

Baroni, M. et Bernardini, S. (2004). BootCaT : Bootstrapping Corpora and Terms from the Web. *Dans Proceedings of the LREC 2004 conference*, pages 1313–1316. Cité pages 19 et 81.

Baroni, M., Bernardini, S., Ferraresi, A. et Zanchetta, E. (2009). The wacky wide web : A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226. Cité page 20.

Baroni, M., Chantree, F., Kilgarriff, A. et Sharoff, S. (2008). Cleaneval : a competition for cleaning web pages. *Dans Proceedings of the Conference on Language Resources and Evaluation (LREC), Marrakech*. Cité page 135.

Baroni, M. et Ueyama, M. (2006). Building general-and special-purpose corpora by web crawling. *Dans Proceedings of the 13th NIJL international symposium, language corpora : Their compilation and application*, pages 31–40. Cité pages 19 et 21.

Baujard, O., Baujard, V., Aurel, S., Boyer, C. et Appel, R. (1998). MARVIN, multi-agent softbot to retrieve multilingual medical information on the Web. *Medical informatics*. Cité page 24.

Baykan, E., Henzinger, M., Marian, L. et Weber, I. (2009). Purely url-based topic classification. *Dans Proceedings of the 18th international conference on World wide web*, WWW '09, pages 1109–1110, New York, NY, USA. ACM. Cité page 102.

Bergman, M. (2001). The deep web : Surfacing hidden value. *The journal of electronic publishing*, 7(1). Cité page 10.

Bergmark, D., Lagoze, C. et Sbityakov, A. (2002). Focused crawls, tunneling, and digital libraries. *Lecture notes in computer science*. Cité page 26.

Bharat, K. et Broder, A. (1998). A technique for measuring the relative size and overlap of public web search engines. *Computer Networks and ISDN Systems*, pages 1–12. Cité page 94.

Blanco, R. et Lioma, C. (2007). Random walk term weighting for information retrieval. *Dans Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 829–830. Cité page 75.

Brants, T., Popat, A., Xu, P., Och, F. et Dean, J. (2007). Large language models in machine translation. *Dans Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, volume 1, pages 858–867. Cité page 90.

Brewington, B. et Cybenko, G. (2000). Keeping up with the changing web. *Computer*, 33(5):52–58. Cité page 28.

Brin, S. (1999). Extracting patterns and relations from the world wide web. *The World Wide Web and Databases*, pages 172–183. Cité page 18.

Broder, A. (2000). Identifying and filtering near-duplicate documents. *Dans Combinatorial Pattern Matching*, pages 1–10. Springer. Cité page 19.

Budzik, J. et Hammond, K. (1999). Watson : Anticipating and contextualizing information needs. *Dans Proceedings of the Annual Meeting-American Society for Information Science*, volume 36, pages 727–740. Cité page 16.

Burner, M. (1997). Crawling Towards Eternity : Building an Archive of the World Wide Web. *Web Techniques Magazine*, 2(5). Cité page 90.

Carpineto, C., Osiński, S., Romano, G. et Weiss, D. (2009). A survey of web clustering engines. *ACM Computing Surveys (CSUR)*, 41(3):17. Cité page 14.

Castillo, C. (2005). *Effective web crawling*. Thèse de doctorat. Cité page 21.

Chakrabarti, S., den Berg, M. V. et Dom, B. (1999). Focused crawling : a new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11-16):1623–1640. Cité pages 23 et 89.

Chakrabarti, S., Punera, K. et Subramanyam, M. (2002). Accelerated focused crawling through online relevance feedback. *Dans Proceedings of the 11th international conference on World Wide Web*, pages 148–159. ACM New York, NY, USA. Cité pages 24, 26, 93, 98, 111, et 114.

Chapelle, O. et Chang, Y. (2011). Yahoo ! learning to rank challenge overview. *Journal of Machine Learning Research-Proceedings Track*, 14:1–24. Cité pages 101, 104, 105, 107, et 110.

Chen, J., Karthik, T. et Subramanian, L. (2010). Contextual information portals. *Dans Proceedings of AAAI Spring Symposium*. Cité page 25.

Cho, J. et Garcia-Molina, H. (2002). Parallel crawlers. *Dans Proceedings of the 11th international conference on World Wide Web*, pages 124–135. ACM. Cité page 22.

Cho, J. et Garcia-Molina, H. (2003a). Effective page refresh policies for web crawlers. *ACM Transactions on Database Systems (TODS)*, 28(4):390–426. Cité page 28.

Cho, J. et Garcia-Molina, H. (2003b). Estimating frequency of change. *ACM Transactions on Internet Technology (TOIT)*, 3(3):256–290. Cité page 28.

Cho, J., Garcia-Molina, H. et Page, L. (1998). Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1-7):161–172. Cité pages 22 et 23.

Choi, B. et Yao, Z. (2005). Web page classification. *Foundations and Advances in Data Mining*, pages 221–274. Cité page 99.

Clarke, C. L., Craswell, N. et Soboroff, I. (2009). Overview of the trec 2009 web track. Cité page 14.

Coffman, E., Liu, Z. et Weber, R. (1997). Optimal robot scheduling for web search engines. *Journal of Scheduling*. Cité page 28.

Cronen-Townsend, S., Zhou, Y. et Croft, W. (2002). Predicting query performance. *Dans Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306. ACM. Cité page 51.

De Bra, P. et Post, R. (1994). Information retrieval in the world-wide web : making client-based searching feasible. *Computer Networks and ISDN Systems*, 27(2):183–192. Cité page 23.

de Groc, C. (2010). Constitution automatique ou semi-automatique de lexiques thématiques en vue de leur utilisation dans un catégoriseur. Mémoire de Master, Université Paris-Sud, Orsay, France. Cité page 50.

DE GROC, C. (2011). Babouk : Focused Web Crawling for Corpus Compilation and Automatic Terminology Extraction. *Dans Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence*, pages 497–498. Cité page 89.

DE GROC, C., COUTO, J., BLANCAFORT, H. et DE LOUPY, C. (2011a). Babouk : exploration orientée du web pour la constitution de corpus et de terminologies. *Dans Traitement Automatique des Langues Naturelles*. Cité page 89.

DE GROC, C. et TANNIER, X. (2012). Experiments on Pseudo Relevance Feedback using Graph Random Walks. *Dans Proceedings of 19th edition of the International Symposium on String Processing and Information Retrieval (SPIRE)*, pages 193–198. Springer. Cité page 75.

DE GROC, C., TANNIER, X. et COUTO, J. (2011b). GrawlTCQ : Terminology and Corpora Building by Ranking Simultaneously Terms , Queries and Documents using Graph Random Walks. *Dans Proceedings of the 6th Workshop on Graph-based Methods for Natural Language Processing*, pages 37–41. Association for Computational Linguistics. Cité page 73.

DE GROC, C., TANNIER, X. et DE LOUPY, C. (2012). Un critère de cohésion thématique fondé sur un graphe de cooccurrences. *Dans Actes de la conférence conjointe JEP-TALN-RECITAL 2012, volume 2 : TALN*, pages 183–195. ATALA/AFCP. Cité page 70.

DE JAGER, D. et BRADLEY, J. (2009). Pagerank : Splitting homogeneous singular linear systems of index one. *Advances in Information Retrieval Theory*, pages 17–28. Cité page 78.

DEAN, J. et GHEMAWAT, S. (2004). Mapreduce : simplified data processing on large clusters. *Dans 6th Symposium on Operating System Design and Implementation (OSDI)*, pages 137—150. Cité page 91.

DÉJEAN, H. et GAUSSIER, E. (2002). Une nouvelle approche à l'extraction de lexiques bilingues à partir de corpus comparables. *Lexicometrica, Alignement lexical dans les corpus multilingues*, pages 1–22. Cité page 6.

DHILLON, I. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. *Dans Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM. Cité page 75.

DIACONIS, P. (1988). Group representations in probability and statistics. *Lecture Notes-Monograph Series*. Cité page 62.

Diaz, F., Lalmas, M. et Shokouhi, M. (2010). From federated to aggregated search. *Dans Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 910–910. ACM. Cité page 12.

Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. et Gori, M. (2000). Focused crawling using context graphs. *Dans Proceedings of the 26th International Conference on Very Large Data Bases*, pages 527–534. Cité pages xi, 23, 24, 25, 93, et 114.

Duh, K. et Kirchhoff, K. (2008). Learning to rank with partially-labeled data. *Dans Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 251–258. ACM. Cité page 97.

Dwork, C., Kumar, R., Naor, M. et Sivakumar, D. (2001). Rank aggregation methods for the web. *Dans Proceedings of the 10th international conference on World Wide Web*, pages 613–622. ACM. Cité page 62.

Edwards, J., McCurley, K. et Tomlin, J. (2001). An adaptive model for optimizing performance of an incremental web crawler. *Dans Proceedings of the tenth international conference on World Wide Web*. Cité page 28.

Evert, S. (2008). A lightweight and efficient tool for cleaning web pages. *Dans Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), Marrakech, Morocco. European Language Resources Association (ELRA).* Cité page 136.

Fan, R., Chang, K., Hsieh, C., Wang, X. et Lin, C. (2008). Liblinear : A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874. Cité page 99.

Farahat, A., LoFaro, T., Miller, J., Rae, G. et Ward, L. (2006). Authority rankings from hits, pagerank, and salsa : Existence, uniqueness, and effect of initialization. *SIAM Journal on Scientific Computing*, 27(4):1181–1201. Cité page 55.

Fekete, J., Wang, D., Dang, N., Aris, A. et Plaisant, C. (2003). Overlaying graph links on treemaps. *Dans IEEE Symposium on Information Visualization Conference Compendium (demonstration)*, volume 5. Cité page 54.

Ferraresi, A., Zanchetta, E., Baroni, M. et Bernardini, S. (2008). Introducing and evaluating ukwac, a very large web-derived corpus of english. *Dans Proceedings of the 4th Web as Corpus Workshop (WAC-4)*, pages 47–54. Cité page 81.

Fetterly, D., Manasse, M. et Najork, M. (2004). Spam , Damn Spam , and Statistics Using statistical analysis to locate spam web pages. *Dans Proceedings of the 7th International Workshop on the Web and Databases : colocated with ACM SIGMOD/PODS 2004*. Cité page 93.

Finn, A., Kushmerick, N. et Smyth, B. (2001). Fact or fiction : Content classification for digital libraries. *Dans DELOS Workshop : Personalisation and Recommender Systems in Digital Libraries*. Cité pages 80, 93, et 135.

Fletcher, W. (2001). Concordancing the web with kwicfinder. *Dans Third North American Symposium on Corpus Linguistics and Language Teaching*, pages 1–16. Cité page 18.

Fletcher, W. H. (2004). Making the Web More Useful as a Source for Linguistic Corpora. *Corpus Linguistics in North America*, (January 2003): 191–205. Cité pages 19 et 93.

Friedman, J. H. (2001). Greedy function approximation : a gradient boosting machine.(english summary). *Ann. Statist*, 29(5):1189–1232. Cité pages 104 et 109.

Ganjisaffar, Y., Caruana, R. et Lopes, C. V. (2011). Bagging gradient-boosted trees for high precision, low variance ranking models. *Dans Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 85–94. ACM. Cité page 105.

Ghani, R. et Jones, R. (2000). Learning a monolingual language model from a multilingual text database. pages 187–193, New York, New York, USA. ACM Press. Cité page 19.

Ghani, R., Jones, R. et Mladenic, D. (2005). Building Minority Language Corpora by Learning to Generate Web Search Queries. *Knowledge and information systems*, 7(1):56–83. Cité pages 19, 73, et 87.

Glover, E., Flake, G., Lawrence, S., Birmingham, W., Kruger, A., Giles, C. et Pennock, D. (2001). Improving category specific web search by learning query modifications. *Dans Applications and the Internet, 2001. Proceedings. 2001 Symposium on*, pages 23–32. IEEE. Cité pages 15 et 18.

Guan, Z., Wang, C., Chen, C., Bu, J. et Wang, J. (2008). Guide focused crawler efficiently and effectively using on-line topical importance estimation. *Dans Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '08*, numéro 3, page 757, New York, New York, USA. ACM Press. Cité page 23.

Gyöngyi, Z. et Garcia-Molina, H. (2005). Web spam taxonomy. *Dans First International Workshop on Adversarial Information Retrieval on the Web*. Cité pages 12 et 93.

Halevy, A., Norvig, P. et Pereira, F. (2009). The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*, 24(2):8–12. Cité page 90.

Hassan, S., Mihalcea, R. et Banea, C. (2006). Random-Walk Term Weighting for Improved Text Classification. *Dans Proceedings of the International Conference on Semantic Computing*, pages 242–249. Cité page 74.

Hauff, C. (2010). *Predicting the effectiveness of queries and retrieval systems*. Thèse de doctorat, Centre for Telematics and Information Technology University of Twente. Cité pages 50 et 51.

Haveliwala, T. (2003). Topic-sensitive pagerank : A context-sensitive ranking algorithm for web search. *Knowledge and Data Engineering, IEEE Transactions on*, 15(4):784–796. Cité page 78.

He, B. et Ounis, I. (2005). A study of the dirichlet priors for term frequency normalisation. *Dans Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 465–471. ACM. Cité page 43.

He, J., Larson, M. et De Rijke, M. (2008). Using coherence-based measures to predict query difficulty. *Advances in Information Retrieval*, pages 689–694. Cité pages 51 et 52.

Herbrich, R., Graepel, T. et Obermayer, K. (1999). Large margin rank boundaries for ordinal regression. *Advances in Neural Information Processing Systems*, pages 115–132. Cité page 104.

Hersovici, M., Jacovi, M., Maarek, Y., Pelleg, D., Shtalhaim, M. et Ur, S. (1998). The shark-search algorithm. an application : tailored web site mapping. *Computer Networks and ISDN Systems*, 30(1-7):317–326. Cité pages 23, 92, 96, 97, et 111.

Järvelin, K. et Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446. Cité page 108.

Jensen, E., Beitzel, S., Grossman, D., Frieder, O. et Chowdhury, A. (2005). Predicting query difficulty on the web by learning visual clues. *Dans Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 615–616. ACM. Cité pages 51 et 52.

Jiang, X., Hu, Y. et Li, H. (2009). A ranking approach to keyphrase extraction. *Dans Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 756–757. ACM. Cité page 97.

Kageura, K. et Umino, B. (1996). Methods of automatic term recognition : A review. *Terminology*, 3(2):259–289. Cité pages 40 et 76.

Khare, R., Cutting, D., Sitaker, K. et Rifkin, A. (2004). Nutch : A flexible and scalable open-source web search engine. Rapport technique, CommerceNet Labs. Cité pages 90 et 91.

Kilgarriff, A. et Grefenstette, G. (2003). Introduction to the Special Issue on the Web as Corpus. *Computational Linguistics*, 29(3):333–347. Cité page 18.

Kluck, M. (2004). Evaluation of cross-language information retrieval using the domain-specific girt data as parallel german-english corpus. *Dans Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC*, pages 1343–1346. Cité page 13.

Kluck, M. et Gey, F. (2001). The domain-specific task of clef - specific evaluation strategies in cross-language information retrieval. *Cross-Language Information Retrieval and Evaluation*, pages 48–56. Cité pages 13 et 35.

Knight, W. R. (1966). A computer method for calculating kendall's tau with ungrouped data. *Journal of the American Statistical Association*, 61(314):436–439. Cité page 108.

Kohlschütter, C., Fankhauser, P. et Nejdl, W. (2010). Boilerplate detection using shallow text features. *Dans Proceedings of the third ACM international conference on Web search and data mining*, pages 441–450. ACM. Cité pages 135 et 136.

Kullback, S. et Leibler, R. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86. Cité page 60.

Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tompkins, A. et Upfal, E. (2000). The web as a graph. *Dans Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–10. ACM. Cité pages 23 et 106.

Lafferty, J. et Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. *Dans Proceedings of the 24th annual international ACM SIGIR conference*, pages 111–119. ACM. Cité pages xii et 75.

Langville, A. et Meyer, C. (2005). A survey of eigenvector methods for web information retrieval. *SIAM review*, pages 135–161. Cité page 55.

Li, H. (2011). *Learning to Rank for Information Retrieval and Natural Language Processing*, volume 12. Morgan & Claypool Publishers. Cité page 104.

Liu, B., Dai, Y., Li, X., Lee, W. et Yu, P. (2003). Building text classifiers using positive and unlabeled examples. *Dans Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 179–186. IEEE. Cité pages 87 et 93.

Liu, T. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331. Cité pages 1 et 97.

Lui, M. et Baldwin, T. (2012). langid. py : An off-the-shelf language identification tool. *Dans Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012), Demo Session, Jeju, Republic of Korea*. Cité page 102.

Lv, Y., Moon, T., Kolari, P., Zheng, Z., Wang, X. et Chang, Y. (2011). Learning to model relatedness for news recommendation. *Dans Proceedings of the 20th international conference on World wide web*, pages 57–66. ACM. Cité page 97.

Maila, M. et Shi, J. (2001). A random walks view of spectral segmentation. *Dans Artificial Intelligence and Statistics*. Cité page 75.

Manevitz, L. et Yousef, M. (2002). One-class svms for document classification. *The Journal of Machine Learning Research*, 2:154. Cité pages 87 et 93.

Manning, C., Raghavan, P. et Schutze, H. (2008). *Introduction to information retrieval*. Cambridge University Press Cambridge. Cité pages 11, 40, et 41.

Mehler, A., Sharoff, S. et Santini, M. (2010). *Genres on the Web : Computational Models and Empirical Studies*, volume 42. Springer Verlag. Cité page 38.

Menczer, F. (1997). Arachnid : Adaptive retrieval agents choosing heuristic neighborhoods for information discovery. *Dans Proceedings of the Fourteenth International Conference on Machine Learning*, pages 227–235. Cité page 24.

Menczer, F., Pant, G. et Srinivasan, P. (2003). Topical web crawlers : Evaluating adaptive algorithms. *ACM Trans. on Internet Technologies*. Cité pages 26 et 111.

Menezes, R. (2004). Interactive focused crawler : Setup, monitoring and control through user feedback. Mémoire de Master, K.R. School of Information Technology, IIT Bombay. Cité page 24.

Metzler, D. et Croft, W. B. (2007). Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274. Cité page 104.

Michael, M., Moreira, J. E., Shiloach, D. et Wisniewski, R. W. (2007). Scale-up x Scale-out : A Case Study using Nutch/Lucene. *Dans IEEE International Parallel and Distributed Processing Symposium*, pages 1–8. Ieee. Cité page 91.

Mihalcea, R. et Tarau, P. (2004). Textrank : Bringing order into texts. *Dans Proceedings of EMNLP*, volume 2004, pages 404–411. Cité pages 54 et 74.

Minkov, E., Cohen, W. et Ng, A. (2006). Contextual Search and Name Disambiguation in Email Using Graphs. *Dans Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 27–34. ACM New York, NY, USA. Cité page 75.

Murdock, V. et Lalmas, M. (2008). Workshop on aggregated search. *Dans ACM SIGIR Forum*, volume 42, pages 80–83. ACM. Cité page 12.

Newman, M. (2004). Analysis of weighted networks. *Physical Review E*, 70(5):056131. Cité page 54.

Nie, Z., Zhang, Y., Wen, J. et Ma, W. (2005). Object-level ranking : Bringing order to web objects. *Dans Proceedings of the 14th international conference on the World Wide Web*, pages 567—-574. ACM. Cité pages 75 et 87.

Ntoulas, A., Najork, M. et Manasse, M. (2006). Detecting spam web pages through content analysis. *Dans Proceedings of the 15th international conference on the World Wide Web*, volume pages, pages 83–92. Cité page 93.

Olston, C. et Najork, M. (2010). Web Crawling. *Foundations and Trends® in Information Retrieval*, 4(3):175–246. Cité page 23.

Oyama, S., Kokubo, T., Ishida, T., Yamada, T. et Kitamura, Y. (2001). Keyword spices : A new method for building domain-specific web search engines. *Dans International Joint Conference in Artificial Intelligence (IJCAI)*, volume 17, pages 1457–1466. Cité pages 15 et 18.

Page, L., Brin, S., Motwani, R. et Winograd, T. (1999). The PageRank Citation Ranking : Bringing Order to the Web. Rapport technique, Stanford InfoLab. Cité pages 55, 56, et 74.

Pandey, S. et Olston, C. (2008). Crawl ordering by search impact. *Dans Proceedings of the international conference on Web search and web data mining*, pages 3–14. ACM. Cité pages 22 et 28.

Pang, B. et Lee, L. (2005). Seeing stars : Exploiting class relationships for sentiment categorization with respect to rating scales. *Dans Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, volume 43, page 115. Cité page 97.

Pant, G. et Menczer, F. (2002). Myspiders : Evolve your own intelligent web crawlers. *Autonomous agents and multi-agent systems*, 5(2):221–229. Cité pages 23 et 24.

Pant, G. et Menczer, F. (2003). Topical crawling for business intelligence. *Research and Advanced Technology for Digital Libraries*, pages 233–244. Cité page 28.

Pant, G. et Srinivasan, P. (2005). Learning to crawl : Comparing classification schemes. *ACM Transactions on Information Systems (TOIS)*, 23(4):430–462. Cité page 98.

Pant, G. et Srinivasan, P. (2006). Link contexts in classifier-guided topical crawlers. *Knowledge and Data Engineering, IEEE Transactions on*, 18(1):107–122. Cité pages 23, 27, 96, 97, 102, et 111.

Pomikálek, J. (2011). *Removing Boilerplate and Duplicate Content from Web Corpora*. Thèse de doctorat, Masaryk University, Brno, Czech Republic. Cité page 135.

Porter, M. (1980). An algorithm for suffix stripping. *Program : electronic library and information systems*, 40(3):211–218. Cité pages 39 et 57.

Qin, J., Zhou, Y. et Chau, M. (2004). Building domain-specific web collections for scientific digital libraries : a meta-search enhanced focused crawling method. *Dans Digital Libraries, 2004. Proceedings of the 2004 Joint ACM/IEEE Conference on*, pages 135–141. IEEE. Cité page 26.

Qin, T., Liu, T., Xu, J. et Li, H. (2010). Letor : A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4):346–374. Cité pages 101 et 105.

Qiu, G., Liu, K., Bu, J., Chen, C. et Kang, Z. (2007). Quantify query ambiguity using odp metadata. *Dans Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 697–698. ACM. Cité page 52.

Ramage, D., Rafferty, A. et Manning, C. (2009). Random walks for text semantic similarity. page 23, Morristown, NJ, USA. Association for Computational Linguistics. Cité page 102.

Rennie, J. et McCallum, A. (1999). Efficient web spidering with reinforcement learning. *Dans Proceedings of the 16th international conference on Machine Learning*. Cité pages 24 et 93.

Renouf, A., Kehoe, A. et Banerjee, J. (2006). Webcorp : an integrated system for web text search. *Language and Computers*, 59(1):47–67. Cité page 18.

Richardson, M. et Domingos, P. (2002). The intelligent surfer : Probabilistic combination of link and content information in pagerank. *Advances in Neural Information Processing Systems*, 2:1441–1448. Cité page 78.

Richardson, M., Prakash, A. et Brill, E. (2006). Beyond pagerank : machine learning for static ranking. *Dans Proceedings of the 15th international conference on World Wide Web*, pages 707–715. ACM. Cité page 97.

Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd édition. Cité page 42.

Riloff, E. et Jones, R. (1999). Learning dictionaries for information extraction by multi-level bootstrapping. *Dans Proceedings of the National Conference on Artificial Intelligence*, pages 474–479. Cité page 18.

Salton, G. et Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523. Cité pages 39 et 40.

Sculley, D. (2009). Large scale learning to rank. *Dans NIPS 2009 Workshop on Advances in Ranking*. Cité page 107.

Sebastiani, F. et Ricerche, C. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34:1–47. Cité pages 73 et 99.

Settles, B. (2010). Active learning literature survey. Rapport technique, University of Wisconsin–Madison. Cité page 87.

Sharoff, S. (2006). Creating general-purpose corpora using automated search engine queries. *M. Baroni, S. Bernardini (eds.) WaCky ! Working papers on the Web as Corpus, Bologna, 2006*, pages 63–98. Cité page 20.

Shokouhi, M. et Si, L. (2011). Federated Search. *Foundations and Trends® in Information Retrieval*, 5(1):1–102. Cité page 12.

Sondhi, P., Chandrasekar, R. et Rounthwaite, R. (2010). Using query context models to construct topical search engines. *Dans Proceedings of the third symposium on Information interaction in context*, pages 75–84. ACM. Cité page 15.

Song, R., Luo, Z., Wen, J., Yu, Y. et Hon, H. (2007). Identifying ambiguous queries in web search. *Dans Proceedings of the 16th international conference on World Wide Web*, pages 1169–1170. ACM. Cité pages 52 et 53.

Spengler, A. et Gallinari, P. (2010). Document structure meets page layout : loopy random fields for web news content extraction. *Dans Proceedings of the 10th ACM symposium on Document engineering*, pages 151–160. ACM. Cité page 135.

Srinivasan, P., Menczer, F. et Pant, G. (2005). A general evaluation framework for topical crawlers. *Information Retrieval*, 8(3):417–447. Cité pages 27, 40, et 111.

Steele, R. (2001). Techniques for specialized search engines. *Proceedings of Internet Computing*, 1:25–28. Cité page 15.

Tang, T., Craswell, N., Hawking, D., Griffiths, K. et Christensen, H. (2006). Quality and relevance of domain-specific search : A case study in mental health. *Information Retrieval*, 9(2):207–225. Cité page 15.

Tang, T., Hawking, D., Sankaranarayana, R., Griffiths, K. et Craswell, N. (2009). Quality-oriented search for depression portals. *Advances in Information Retrieval*, pages 637–644. Cité page 15.

Wang, R. et Cohen, W. (2007). Language-independent set expansion of named entities using the web. *Dans Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 342–350. Cité pages 74, 75, 76, et 78.

Wermter, J. et Hahn, U. (2006). You can't beat frequency (unless you use linguistic knowledge). *Dans Annual Meeting-Association for Computational Linguistics*, volume 44, page 785. Cité pages 50 et 87.

Wu, Q., Burges, C. J., Svore, K. M. et Gao, J. (2010). Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270. Cité page 104.

Yu, H., Han, J. et Chang, K. (2004). PEBL : Web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):70–81. Cité pages 93 et 94.

Yu, H., Huang, F. et Lin, C. (2011). Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1):41–75. Cité page 99.

ÉVALUATION DU NETTOYAGE DE PAGES WEB

A

En 2008, la campagne d'évaluation CLEANEVAL (Baroni et coll., 2008) a fédéré les recherches sur le nettoyage de pages Web. Toutefois, cette campagne n'a pas été reproduite et les méthodes évaluées après cette dernière n'ont pas toujours été comparées sur un même corpus. Afin de sélectionner la meilleure méthode de nettoyage de page pour notre problématique, nous avons réutilisé le corpus CLEANEVAL [1] pour comparer les meilleures méthodes existantes.

Nous nous intéressons plus précisément aux méthodes de nettoyage fonctionnant sur des pages Web quelconques, par opposition aux méthodes de nettoyage de page se spécialisant sur les pages d'actualités (Kohlschütter et coll., 2010; Spengler et Gallinari, 2010). Nous avons sélectionné quatre algorithmes parmi les plus performants : *(i)* BodyTextExtraction (BTE) de Finn et coll. (2001) qui a obtenu les meilleurs résultats dans le cadre de la campagne CLEANEVAL. Ce dernier sélectionne la portion de texte la plus large contenant le moins de balises HTML ; *(ii)* Boilerpipe (Kohlschütter et coll., 2010) qui emploie un arbre de décision qui s'appuie principalement sur la densité du texte (ratio entre le nombre de mots et le nombre de lignes), de liens et de balises dans, avant, et après le bloc à classer. Il a montré de très bonnes performances sur un corpus de pages d'actualités (*Google News*) et des performances état de l'art sur le corpus CLEANEVAL ; *(iii)* jusText (Pomikálek, 2011), un algorithme heuristique fonctionnant en deux phases : la première phase a pour objectif d'annoter chaque bloc comme bon, mauvais ou « quasi bon ». Puis une seconde phase tenant compte du contexte des blocs (blocs précédent et suivant) affine les annotations ; *(iv)* Readability [2] qui repose sur un ensemble d'heuristiques et est utilisé par le grand public comme *plug-in* de navigateurs Web (Mozilla Firefox, Apple Safari) pour faciliter la lecture de pages Web. Nous ajoutons à ces quatre algorithmes une *baseline* correspondant à la stratégie de nettoyage utilisée jusqu'à présent et renvoyant tout le texte de la page par simple suppression des balises.

1. Nous avons utilisé la version de Pomikálek (2011) où deux fichiers (numéros 103 et 250) ont été supprimés, car les annotations produites pour ces derniers étaient invalides.
2. http://lab.arc90.com/experiments/readability/

Méthode	Précision	Rappel	F_1-Mesure
Baseline	86.75	**95.46**	90,90
Boilerpipe	95.47	90,12	92.72
BTE	92.81	93.00	**92.80**
jusText	**96.67**	84.82	90,36
Readability	95.68	83.95	89.43

Tableau A.1 : Évaluation de différentes méthodes de nettoyage de pages sur le corpus CLEANEVAL

Nous avons utilisé l'outil d'évaluation de Evert (2008) qui mesure l'écart en nombre de mots entre les fichiers nettoyés automatiquement et la référence constituée manuellement. Nous présentons les valeurs de précision, rappel et F_1-mesure micro moyennée sur l'ensemble du corpus au Tableau A.1. Notons que la *baseline* n'obtient pas un rappel de 100 %. Cela avait déjà été constaté par Evert qui suggère que ce phénomène est dû à des problèmes d'encodage et l'apparition de texte de formatage ou caché. De plus, la simple suppression des balises est en elle même déjà un challenge tant les pages Web sont bruitées (balises de script, balises de fin de page en milieu de page, ...). Nous constatons que la *baseline* produit une F_1-Mesure relativement haute et proche des autres méthodes, ce qui avait déjà été constaté par Kohlschütter et coll. (2010). La méthode jusText offre la meilleure précision, mais un rappel plus faible. Les algorithmes Boilerpipe et BTE fournissent les meilleures F_1-Mesure et assurent un certain équilibre entre précision et rappel. Nous choisissons finalement l'algorithme BTE pour la suite de nos expériences. Ce denier offre une précision plus faible que Boilerpipe, mais son fonctionnement et son implémentation simples ainsi que sa rapidité d'exécution en font une méthode de choix.

B

PUBLICATIONS RÉALISÉES DURANT LA THÈSE

CHAPITRES D'OUVRAGES :

– COUTO, J., DE GROC, C., GUÉGAN, M. ET DE LOUPY, C. (2011). La navigation à facettes dans les moteurs de recherche. *Dans* GRIVEL, L., ÉDITEUR : *La recherche d'information en contexte : outils et usages applicatifs*. HERMÈS SCIENCE PUBLICATIONS.

CONFÉRENCES INTERNATIONALES :

– DE GROC, C., TANNIER, X. ET COUTO, J. (2011c). GrawlTCQ : Terminology and Corpora Building by Ranking Simultaneously Terms , Queries and Documents using Graph Random Walks. *Dans Proceedings of the 6th Workshop on Graph-based Methods for Natural Language Processing*, pages 37–41. Association for Computational Linguistics.

– DE GROC, C. ET TANNIER, X. (2012B). EXPERIMENTS ON PSEUDO RELEVANCE FEEDBACK USING GRAPH RANDOM WALKS. *Dans Proceedings of 19th edition of the International Symposium on String Processing and Information Retrieval (SPIRE)*, PAGES 193–198.

– ALONSO, A., BLANCAFORT, H., DE GROC, C., MILLION, C. ET WILLIAMS, G. (2012). METRICC : HARNESSING COMPARABLE CORPORA FOR MULTILINGUAL LEXICON DEVELOPMENT. *Dans Proceedings of the 15th EURALEX International Congress*, PAGES 389–403.

– FERREZ, R., DE GROC, C. ET COUTO, J. (2012). SELF-SUPERVISED PRODUCT FEATURE EXTRACTION USING A KNOWLEDGE BASE AND VISUAL CLUES. *Dans Proceedings of the 8th International Conference on Web Information Systems and Technologies*, PAGES 643–652.

– Ferrez, R., de Groc, C. et Couto, J. (2013). Mining Product Features from the Web : a Self-Supervised Approach. *Lecture Notes in Business Information Processing (LNBIP)*, volume 140, pages 296–311. Version étendue et révisée de (Ferrez et coll., 2012)

conférences nationales :

– de Groc, C., Tannier, X. et de Loupy, C. (2012a). Un critère de cohésion thématique fondé sur un graphe de cooccurrences. *Dans Actes de la conférence conjointe JEP-TALN-RECITAL 2012, volume 2 : TALN*, pages 183–195. ATALA/AFCP.

démonstrations :

– de Groc, C. (2011a). Babouk : Focused Web Crawling for Corpus Compilation and Automatic Terminology Extraction. *Dans Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence*, pages 497–498.

– de Groc, C., Couto, J., Blancafort, H. et de Loupy, C. (2011b). Babouk : exploration orientée du web pour la constitution de corpus et de terminologies. *Dans Traitement Automatique des Langues Naturelles*.

notes de lecture :

– de Groc, C. (2011c). Data-Intensive Text Processing with MapReduce. Jimmy Lin and Chris Dyer. *Traitement Automatique des Langues*, 51(3): 159–162.