Apprendre à ordonner la frontière de crawl pour le crawling orienté

Clément de Groc* — **Xavier Tannier****

* *Syllabs, Paris, France*
** *LIMSI-CNRS, Univ. Paris-Sud, Orsay, France*

RÉSUMÉ. *Le crawling orienté consiste à parcourir le Web au travers des hyperliens en orientant son parcours en direction des pages pertinentes. Pour cela, ces crawlers ordonnent leurs téléchargements suivant une stratégie d'ordonnancement. Dans cet article, nous proposons d'apprendre cette fonction d'ordonnancement à partir de données annotées. Une telle approche nous permet notamment d'intégrer un grand nombre de traits hétérogènes et de les combiner. Nous décrivons une méthode permettant d'apprendre une fonction d'ordonnancement indépendante du domaine pour la collecte thématique de documents. Nous évaluons notre approche sur de "longs" crawls de 40 000 documents sur 15 thèmes différents issus de l'OpenDirectory, et montrons que notre méthode permet d'améliorer la précision de près de 10% par rapport à l'algorithme Shark Search. Enfin, nous discutons les avantages et inconvénients de notre approche, ainsi que les pistes de recherche ouvertes.*

ABSTRACT. *Focused crawling consists in searching and retrieving a set of documents relevant to a specific domain of interest from the Web. Such crawlers prioritize their fetches by relying on a crawl frontier ordering strategy. In this article, we propose to learn this ordering strategy from annotated data using learning-to-rank algorithms. Such approach allows us to cope with tunneling and to integrate a large number of heterogeneous features to guide the crawler. We describe a novel method to learn a domain-independent ranking function for topical Web crawling. We validate the relevance of our approach on "large" crawls of 40,000 documents on a set of 15 topics from the OpenDirectory, and show that our approach provides an increase in precision (harvest rate) of up to 10% compared to a baseline Shark Search algorithm. Finally, we discuss future leads regarding the application of learning-to-rank to focused Web crawling.*

MOTS-CLÉS : *Crawling orienté, Apprentissage de fonction d'ordonnancement, Recherche d'Information sur le Web.*

KEYWORDS: *Focused Crawling, Learning-to-Rank, Web Information Retrieval.*

CORIA 2014, pp. 205–220, Nancy, 18-21 mars 2014

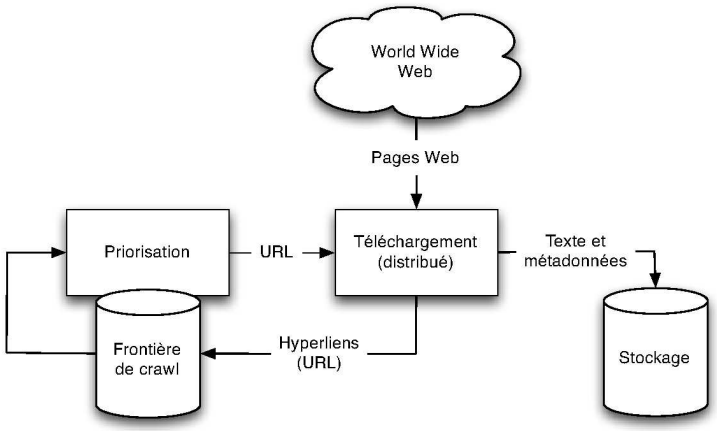

Figure 1. *Processus itératif de crawling du Web.*

1. Introduction

Les *crawlers* Web sont des programmes parcourant le Web au travers des hyperliens entre documents. Le *crawler* démarre à partir d'une liste d'URL dites amorces, et itérativement télécharge et analyse de nouvelles pages Web pour découvrir de nouveaux liens (figure 1, inspirée de (Castillo, 2005)). La liste d'URL découvertes jusqu'alors, appelée frontière de *crawl* (Cho *et al.*, 1998), croît de manière très rapide et les stratégies d'ordonnancement de cette frontière, permettant de prioriser les téléchargements, sont donc essentielles pour télécharger les pages les plus pertinentes d'abord.

Un *crawler* orienté (Chakrabarti *et al.*, 1999) 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 al.*, 1999). De plus, ces derniers permettent un rafraîchissement de l'index du moteur de recherche beaucoup plus fréquent (Diligenti *et al.*, 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 trouver un maximum de pages pertinentes tout en minimisant le nombre de pages visitées qui ne sont pas en rapport avec le domaine ciblé. Les *crawlers* orientés sont aujourd'hui utilisés dans de nombreuses tâches telles que l'intelligence économique, la recherche d'information spécialisée, le *Web-as-corpus* (Baroni et Ueyama, 2006) et la création de portails d'informations (Chen *et al.*, 2011).

Dans cet article, nous proposons d'apprendre une fonction d'ordonnancement thématique à partir de données annotées automatiquement. Tout d'abord, plusieurs *crawls* sont réalisés et annotés automatiquement à l'aide de catégoriseurs de pages Web robustes entraînés sur le second niveau de l'OpenDirectory[1]. Puis, ces *crawls* sont transformés en une liste ordonnée de liens représentant un parcours pseudo-optimal, avant d'être fournis en entrée d'un algorithme d'apprentissage de fonctions d'ordonnancement. La fonction d'ordonnancement résultante est enfin appliquée au sein d'un *crawler* orienté pour collecter des documents sur de nouveaux thèmes. Nous pensons que l'apprentissage de fonctions d'ordonnancement offre notamment l'avantage de fournir un cadre théorique pour exploiter un grand nombre de traits hétérogènes et les combiner.

Pour valider notre approche, nous avons réalisé une série d'expériences sur 15 thèmes issus du second niveau de l'OpenDirectory. Nous comparons notre approche à l'algorithme Shark Search sur ce que les précédents travaux considèrent comme de "longs" *crawls* (Menczer *et al.*, 2004) de 40 000 documents. Nous montrons que notre méthode permet d'améliorer la précision (aussi appelée *harvest rate* dans ce domaine) de près de 10%. Par ailleurs, nous montrons que notre approche offre des performances plus stables sur l'ensemble des thèmes.

La suite de cet article est structurée comme suit : la section 2 présente un bref état de l'art sur le *crawling* orienté. La section 3 décrit une méthode pour inférer une fonction d'ordonnancement à partir de données de *crawl*. La section 4 est dédiée à l'évaluation de l'approche proposée en conditions réelles sur un panel de 15 thèmes. Enfin, nous discutons les avantages et inconvénients de notre approche, ainsi que les travaux futurs en section 5.

2. Travaux liés

La pierre angulaire des *crawlers* orientés est la stratégie d'ordonnancement de la frontière de *crawl*. Cette stratégie vise à télécharger un maximum de pages pertinentes tout en minimisant le nombre de pages non pertinentes visitées. Dans la littérature existante, ces stratégies sont séparées en trois générations (Olston et Najork, 2010).

Les premiers *crawlers* orientés ont été basés sur l'hypothèse de localité thématique (Davison, 2000), soit que des liens extraits de documents pertinents ont de grandes chances de mener à d'autres documents pertinents (Cho *et al.*, 1998). Un exemple de *crawler* exploitant une telle stratégie est Fish Search (De Bra et Post, 1994) qui classe une page comme pertinente ou non pertinente à l'aide d'expressions régulières, puis télécharge en priorité les liens des pages jugées pertinentes.

Les stratégies de seconde génération ont tenté d'apporter plus de précision en considérant les liens sortants d'une page de manière non uniforme. Ainsi, sur une même page, le *crawler* favorise les liens qu'il juge les plus pertinents en analy-

1. `http://www.dmoz.org`

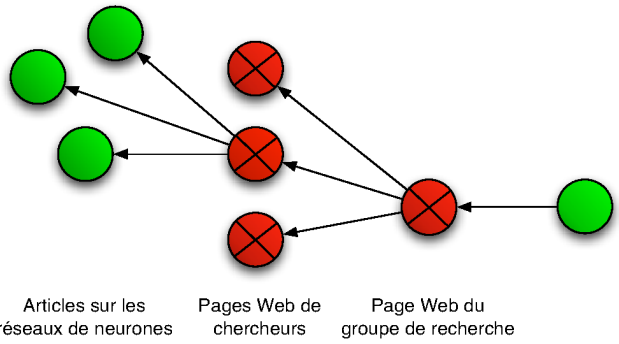

Figure 2. *Exemple d'effet tunnel qu'un crawler orienté devrait détecteur pour télécharger des articles pertinents sur les réseaux de neurones.*

sant le texte d'ancrage ainsi que le texte autour du lien (son *contexte*) (Chakrabarti *et al.*, 2002 ; Hersovici *et al.*, 1998 ; Pant et Srinivasan, 2006). Cette génération de *crawlers* s'appuie également sur des catégoriseurs plus avancés calculant une similarité ou opérant une régression logistique pour évaluer la pertinence des pages. Shark Search est l'une de ces techniques (Hersovici *et al.*, 1998, figure 2) et modélise la pertinence des liens par une combinaison linéaire de la pertinence de la page parente, du texte d'ancrage et du contexte du lien. Dans nos travaux, nous utilisons une version légèrement modifiée de Shark Search : là où l'algorithme original jugeait de la pertinence d'une page à l'aide d'une similarité cosinus sur un petit ensemble de termes fournis par un utilisateur (Hersovici *et al.*, 1998, figure 2, item 1), nous appliquons une catégorisation thématique plus robuste (décrite § 3.2).

Enfin, la troisième génération de *crawlers* s'est attaquée à la modélisation de phénomènes plus complexes tels que l'adaptation du *crawler* au cours du *crawl* (Rennie et McCallum, 1999) ou l'effet tunnel (Bergmark *et al.*, 2002 ; Diligenti *et al.*, 2000). Ce dernier peut être résumé comme suit : le chemin pour aller d'une page pertinente à d'autres pages pertinentes peut inclure des pages non pertinentes. Diligenti et coll. (2000) fournissent un exemple relativement clair d'effet tunnel (figure 2) : un *crawler* orienté ayant pour but de collecter des articles scientifiques sur les réseaux de neurones pourrait avoir à passer par des pages d'université, de groupes de recherche et de chercheurs avant enfin d'aboutir aux articles convoités. Pour prendre en compte ce phénomène, Diligenti et coll. (2000) collectent des informations sur le voisinage des URL amorces dans le graphe de liens du Web à l'aide de requêtes à un moteur de recherche (*backlinks*). Puis, ils entraînent un catégoriseur pour chaque couche de leur graphe de contexte, ce qui leur permet enfin de prédire, pour une page donnée, à quelle distance se trouvent les prochaines pages pertinentes.

Au vu de ces travaux, nous pouvons tirer plusieurs conclusions : *(i)* les approches de seconde génération fusionnent plusieurs indices quant à la pertinence

d'un lien, mais utilisent des combinaisons linéaires où les poids sont fixés manuellement (Hersovici *et al.*, 1998 ; Pant et Srinivasan, 2006) ; *(ii)* peu de traits ont été explorés pour estimer la pertinence d'un lien ; *(iii)* l'effet tunnel, bien que difficile à prendre en compte et à évaluer, semble une piste prometteuse.

3. Apprentissage de fonction d'ordonnancement pour le *crawling* orienté

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 documents en réponse à une requête (apprentissage de fonctions d'ordonnancement). 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 al.*, 2011) ou l'analyse de sentiment (Pang et Lee, 2005).

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 (hyper-)liens sont les documents annotés.

Une première approche pour apprendre cette fonction d'ordonnancement serait simplement de réaliser plusieurs *crawls* thématiques, d'annoter les documents par pertinence et d'apprendre une fonction d'ordonnancement qui téléchargerait les liens vers les pages les plus pertinentes d'abord. Cependant, nous pensons qu'il est possible de capturer certains aspects de l'effet tunnel en ne favorisant pas uniquement les pages pertinentes directement accessibles, mais en favorisant également les liens vers des pages qui mèneront le *crawler* à d'autres pages pertinentes par la suite.

Nous proposons une approche auto supervisée en quatre étapes (figure 3, inspirée de (Li, 2011)) :

1) Sélectionner un thème et *crawler* le Web en quête de documents sur ce thème ;

2) Catégoriser les documents découverts à l'aide de catégoriseurs robustes ;

3) Convertir le graphe de liens du *crawl* et les pages annotées en une liste ordonnée de paires lien/score ;

4) Extraire un ensemble de traits pour chaque lien et apprendre une fonction d'ordonnancement à partir des paires traits/score.

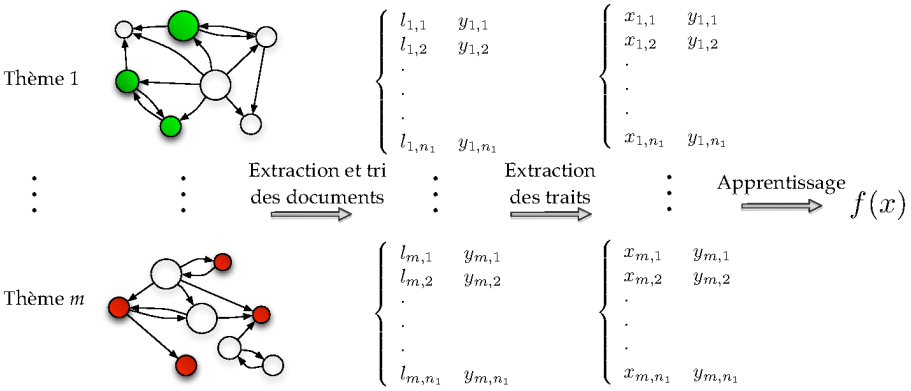

Figure 3. *Apprentissage d'une fonction d'ordonnancement à partir d'un ensemble de thèmes prédéfinis.*

Notre approche vise à apprendre une fonction d'ordonnancement indépendante du domaine. Mentionnons que cette hypothèse est discutable, car, par exemple, Diligenti et coll. (2000) apprennent des classifieurs dépendant du domaine pour prédire les tunnels. Nous pensons que notre approche pourrait également être appliquée pour apprendre des fonctions d'ordonnancement dépendantes du thème. Cependant, il semble plus attractif de tenter d'apprendre une fonction d'ordonnancement indépendante du thème une fois pour toutes.

3.1. *OpenDirectory*

Dans nos travaux, nous utilisons l'OpenDirectory (ODP) comme base de documents thématiques de référence[2] et plus particulièrement pour démarrer le *crawl* (URL amorces) et pour entraîner des catégoriseurs automatiques de pages Web qui nous permettront d'étiqueter les documents téléchargés. Notons que plusieurs travaux antérieurs utilisaient déjà cette ressource de manière analogue ainsi que pour l'évaluation (Chakrabarti *et al.*, 1999 ; Pant et Srinivasan, 2006). L'OpenDirectory est un thesaurus classant les sites Web dans un arbre de catégories. La majeure partie du répertoire est organisé thématiquement et les catégories non thématiques *News* et *Regional* n'ont pas été conservées. L'OpenDirectory ne fournit pas le contenu des sites Web, mais uniquement leur adresse (URL). Par conséquent, nous avons téléchargé les pages Web correspondant aux entrées anglaises de l'OpenDirectory, soit un total de 2 339 125 pages HTML sur 2 463 769 URL initiales. Les pages Web manquantes sont

2. Nous avons utilisé une version du répertoire datant du 15 septembre 2011 qui n'inclut pas les catégories `Adult` et `Kids_and_Teens`.

distribuées de manière relativement uniforme sur l'ensemble des catégories et n'ont donc pas d'influence sur la suite de nos travaux, statistiquement parlant.

Nous nous limitons au second niveau de l'OpenDirectory qui inclue plus de 300 catégories relativement précises tout en offrant de bonnes performances de catégorisation (Liu *et al.*, 2005). Pour chaque catégorie du second niveau, nous avons entraîné un catégoriseur binaire basé sur un modèle de régression logistique *via* LIBLINEAR (Fan *et al.*, 2008). Le classifieur tient compte de la structure de la page Web en considérant les différentes parties de la page comme autant de sacs de mots (titre, corps, texte d'ancrage, meta informations et entêtes). Seules les fréquences des termes normalisées sont utilisées. Nous avons évalué et ajusté[3] les paramètres de chaque classifieurs par double validation croisée. En moyenne, nous obtenons une précision de 0,83 ($\pm 0,10$), un rappel de 0,56 ($\pm 0,15$) et une F_1-mesure de 0,66 (± 0.14). Ces performances étant trop faibles pour nous permettre de générer nos données d'apprentissage, nous allons devoir nous contenter d'un sous-ensemble des catégories du second niveau offrant de meilleures performances de classification.

3.2. *Données de crawl*

La première étape de notre approche auto supervisée est la collecte de données de *crawl* pour un ensemble de thèmes. Nous avons sélectionné manuellement 15 thèmes relativement divers et sur lesquels nos catégoriseurs fournissaient des performances hautes : `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`. Sur ces 15 catégories, la validation croisée fournit les performances suivantes en moyenne : une précision de $0,94$ ($\pm 0,03$), un rappel de $0,75$ ($\pm 0,07$) et une F_1-mesure de $0,83$ ($\pm 0,05$).

Pour produire une quantité suffisamment large de pages pertinentes, non pertinentes et de tunnels, nous avons défini une stratégie de *crawl* dédiée : pour chaque thème, 20 URL amorces sont choisies et fournies en entrée du crawler. Puis le *crawler* itère téléchargements (300 URL) et extractions de nouveaux liens. Lors de l'analyse d'une page, les liens sont extraits et pondérés par l'algorithme Shark Search. Puis, nous choisissons aléatoirement 10 liens sortant en suivant la distribution des poids. Nous pensons que cette approche fournit un ensemble représentatif de liens et que ces liens permettent d'estimer correctement la propension d'aboutir à des pages pertinentes. Les *crawls* sont arrêtés lorsque 10 000 pages ont été téléchargées. Les pages téléchargées pour chaque *crawl* sont ensuite annotées par nos catégoriseurs thématiques et tous les liens menant vers des documents non téléchargés sont omis.

3. Nous avons ajusté deux paramètres : le paramètre de coût/régularisation ainsi que les poids de chaque classe.

Tableau 1. *Échelle de pertinence utilisée pour étiqueter les liens.*

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

3.3. *Des graphes de crawl aux listes de liens ordonnées*

Nous disposons à présent de 15 crawls de 10 000 documents dont les pages sont annotées en fonction de leur pertinence. Pour passer de ces données aux listes de liens ordonnées par ordre de visite pseudo-optimale, nous définissons une échelle de 4 valeurs favorisant les pages pertinentes ainsi que les pages menant à d'autres pages pertinentes à une distance maximale de 3 liens (tableau 1). Bien qu'il semble tentant d'utiliser une échelle de valeurs plus précise, tenant compte par exemple de la distance des pages ou du nombre de pages pertinentes accessibles, nous avons choisi de rester sur une échelle de taille faible à l'instar des travaux précédents (Chapelle et Chang, 2011, tableau 1).

Au final, chaque *crawl* est composé de 360 000 liens en moyenne annotés d'un score de pertinence, soit un total de 5,4 millions de liens dans le jeu de données complet. Le degré sortant moyen d'une page est 189 ($\pm$77), bien loin des estimations de Kumar et coll. qui avaient observé un degré moyen sortant de 7,2 en 2000. Notons qu'avec un tel degré sortant, la frontière de *crawl* contiendra déjà plus d'un million d'URL après seulement 5 000 pages téléchargées, ce qui montre une nouvelle fois l'importance de définir une bonne stratégie d'ordonnancement pour le *crawling* orienté.

3.4. *Extraction de traits*

En recherche d'information, les traits utilisés pour l'apprentissage de fonctions d'ordonnancement sont généralement classés en trois familles (Qin *et al.*, 2010), qui une fois adaptées à notre problématique deviennent : *(i)* les traits relatifs à un thème (notés T) ; *(ii)* les traits relatifs à un lien (notés L) ; *(iii)* les traits relatifs à un thème et un lien (notés T–L).

Par ailleurs, lorsque nous évaluons la pertinence de suivre un lien, nous considérons plusieurs sources d'information : le titre, le corps et la zone de contenu informatif (fournie par l'algorithme BodyTextExtraction[4] (Finn *et al.*, 2001)), l'URL cible, le texte d'ancrage, le contexte (fenêtres de 10, 20 et 40 mots (Pant et Srinivasan, 2006)),

4. Cet algorithme vise à supprimer les zones des pages Web dédiées principalement au formatage telles que l'entête, le pied de page ou les publicités.

le premier bloc englobant le lien (premier noeud DOM de type bloc ancêtre du lien[5]) et les pages ancêtres de la page courante.

La liste complète des 35 traits utilisés est donnée au tableau 2. La position du lien dans le code HTML est un ratio entre la ligne où le lien apparaît dans le code HTML et le nombre total de lignes. La distance en terme de répertoires est simplement la différence entre le nombre de barres obliques dans l'URL de la page courante et l'URL cible. La similarité n-grammes est une similarité cosinus basée sur des segments de 4 à 8 caractères (Baykan *et al.*, 2009). La similarité sac de mots est une similarité cosinus *ntc* (Manning *et al.*, 2008). La détection de langue est opérée *via* un classifieur Bayesien naïf sur des séquences de 1 à 4 caractères. Notons enfin que de nombreux autres traits pourraient être ajoutés à ces 35 premiers. À titre d'exemple, nous avons également tenté d'intégrer plusieurs méthodes pour extraire le contexte des liens (à la fois des fenêtres de taille fixe ou des contextes dérivés de l'arbre DOM (Pant et Srinivasan, 2006)), d'ajouter des traits visuels obtenus après rendu de la page, ou d'employer une mesure de similarité sémantique (Ramage *et al.*, 2009). Cependant, ces traits étaient généralement trop coûteux à calculer, les rendant inutilisables sur nos larges jeux de données.

3.5. *Algorithmes d'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) : *(i) pointwise*, apprendre à prédire la pertinence d'entrées ; *(ii) pairwise*, apprendre à ordonner des paires d'entrées ; *(iii) listwise*, apprendre à ordonner une 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 avons choisi d'évaluer quatre approches :

1) GBRT (Friedman, 2001), approche *pointwise* par régression non linéaire ;

2) RankSVM (Herbrich *et al.*, 1999), approche *pairwise* par classification linéaire ;

3) Coordinate Ascent (Metzler et Croft, 2007), approche *listwise* linéaire ;

4) LambdaMART (Wu *et al.*, 2010), approche *listwise* non linéaire.

Ces approches sont réparties sur les trois familles d'algorithmes d'apprentissage de fonctions d'ordonnancement et nous paraissent pertinentes pour les raisons suivantes : RankSVM a obtenu des performances intéressantes lors du challenge LETOR (Qin *et al.*, 2010) et semble donc une approche *pairwise* de choix. Les arbres de décision boostés (*Gradient Boosted Regression Trees* ou GBRT) sont une approche

5. Voir `http://www.w3.org/TR/html401/struct/globcoll.html`, section 7.5.3.

Tableau 2. *Traits extraits pour chaque lien.*

	Type	Trait	Source
1	L	Profondeur / Nombre de barres obliques	URL
2	L		URL
3	L	Nombre de caractères	URL
4	L	Lien relatif ou absolu	URL
5	L	Lien interne ou externe	URL
6	L	Distance en terme de répertoires (barres obliques)	URL
7	L	Position du lien dans le code HTML	URL
8	L	Position normalisée du lien dans l'arbre DOM	URL
9	L	Lien dans la zone de contenu informatif	URL
10	L	Nombre de mots	URL
11	L		URL page courante
12	L		Texte d'ancrage
13	L		Titre de la page
14	L		Corps de la page
15	L		Contenu informatif
16	L		Bloc englobant
17	L	Nombre de liens	Page courante
18	L		Bloc englobant
19	L	Probabilité que la page soit en anglais	Corps de la page
20	T–L	Similarité sur les n-grammes de caractères	URL
21	T–L		URL page courante
22	T–L	Similarité sac de mots	URL
23	T–L		URL page courante
24	T–L		Texte d'ancrage
25-28	T–L		Contexte du lien
29	T–L		Titre de la page
30	T–L		Corps de la page
31	T–L		Contenu informatif
32	T–L		Bloc englobant
33	T–L	Régression logistique	Page courante
34	T–L		Page parente
35	T–L		Page grand-parente

standard pour l'apprentissage de fonctions d'ordonnancement et LambdaMART, également basé des arbres de décision boostés, a obtenu les meilleures performances au 1er défi du challenge Yahoo ! (Chapelle et Chang, 2011). Enfin, *Coordinate Ascent* (CA) est un algorithme *listwise* simple et efficace pour apprendre une fonction d'ordonnancement.

Notons que, comparativement aux jeux de données standards en apprentissage de fonctions d'ordonnancement (Chapelle et Chang, 2011, tableau 1), notre jeu de données possède un très faible nombre de "requêtes" (15 thèmes), mais un grand nombre de "documents" (5,4 M liens). Ce nombre de liens élevé nous impose d'échantillonner nos données pour entraîner les algorithmes d'apprentissage. Aslam et coll. (2009)

Tableau 3. *Importance relative des 10 meilleurs traits pour l'algorithme GBRT.*

Rang	Trait	Poids (%)
1	Probabilité que la page soit en anglais	6,7%
2	Régression logistique sur la page grand-parente	6,7%
3	Similarité sac de mots sur un contexte de 20 mots	4,7%
4	Lien dans la zone de contenu informatif	4,1%
5	Similarité n-grammes sur l'URL	3,8%
6	Longueur du titre de la page courante	3,3%
7	Similarité sac de mots sur un contexte de 10 mots	3,1%
8	Régression logistique sur la page courante	2,7%
9	Similarité sac de mots sur l'URL cible	2,6%
10	Similarité sac de mots sur l'URL courante	2,0%

ont montré que plusieurs algorithmes d'apprentissage de fonctions d'ordonnancement entrainés sur un sous-ensemble aléatoire (stratégie dénommée *InfAP*) de LETOR obtenaient une précision moyenne équivalente aux mêmes algorithmes entrainés sur son intégralité. Ainsi, pour les algorithmes GBRT, CA et LambdaMart, nous sélectionnons aléatoirement un échantillon de 500 000 entrées (15% du jeu de données complet)[6]. Concernant RankSVM, qui nécessite la construction de paires d'entrées, nous appliquons l'approche stochastique de Sculley (2009) et construisons un sous-ensemble d'un million de paires d'entrées[6].

Nous comparons ces différents algorithmes en validation croisée à l'aide de la précision moyenne (MAP) et du gain cumulé normalisé (NDCG). Nos résultats montrent des valeurs de NDCG très hautes ($0,98$ à $0,99$) qui nous amènent à penser que les différents algorithmes identifient les liens pertinents avec une relative facilité. La MAP est pour sa part maximale pour les algorithmes *listwise* LambdaMart ($0,89$) et CA ($0,88$), suivis par l'algorithme *pointwise* GBRT ($0,82$) et enfin RankSVM ($0,80$).

En marge de ces résultats, l'algorithme GBRT opère intrinsèquement une sélection d'attributs (Friedman, 2001) qu'il nous semble intéressant d'étudier. Le tableau 3 présente l'*importance relative* des traits obtenue en moyenne après une validation croisée à 5 plis. Les traits les plus discriminants tels que la probabilité que le document soit en anglais obtiennent une importance maximale. Contrairement à Pant et Srinivasan (2006) qui obtiennent de meilleurs résultats en utilisant des contextes de 40 mots, l'algorithme GBRT assigne des poids plus importants aux contextes de 20 et 10 mots. Par ailleurs, nous pouvons constater un poids remarquablement haut assigné au fait que le lien soit dans la zone de contenu informatif, trait qui n'avait jamais été étudié dans les travaux antérieurs.

6. Ce nombre a été choisi pour réduire le temps d'apprentissage sans diminuer les performances.

4. Évaluation

Nous évaluons à présent la pertinence des algorithmes d'apprentissage de fonctions d'ordonnancement pour une tâche concrète de *crawling*. Nous nous limitons dans cette évaluation à l'algorithme LambdaMART ayant obtenu les meilleurs résultats en validation croisée. Nous comparons la stratégie proposée *Learning-to-Rank* avec un parcours en largeur et la stratégie (modifiée) Shark Search.

Le comportement 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 al.*, 1998, fig. 2). Dans nos expériences, nous utilisons les valeurs $d = 0,5$ et $b = 0,8$ à l'instar d'Hersovici et coll. Comme Menczer et coll. (2004), nous fixons g à $0,1$. 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) qui s'appuient sur l'apparition des documents de l'OpenDirectory durant le *crawl* ;

– Appliquer un catégoriseur thématique entraîné sur l'OpenDirectory (Chakrabarti *et al.*, 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 optons pour la seconde solution. Bien que l'utilisation d'un catégoriseur automatique implique une référence imparfaite, notre objectif est de 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 quinze nouvelles catégories, toujours extraites du second niveau de l'OpenDirectory : `Arts/Bodyart`, `Business/Aerospace_and_Defense`, `Business/Real_Estate`, `Computers/Robotics`, `Games/Card_Games`, `Games/Gambling`, `Health/Nursing`, `Health/Pharmacy`, `Home/Gardening`, `Recreation/Climbing`, `Shopping/Tools`, `Society/Death`, `Society/Law`, `Sports/Equestrian`, `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 40 000 documents. Une fois téléchargés, les 40 000 documents sont ordonnés en fonction de leur date de téléchargement et la mesure de précision à N est calculée. Une moyenne des mesures est ensuite calculée sur l'ensemble des 15 catégories.

Les résultats sont présentés à la figure 4 et au tableau 4. Comme observé dans les travaux précédents (Diligenti *et al.*, 2000 ; Rennie et McCallum, 1999), le parcours en largeur dérive très rapidement vers des documents non pertinents malgré l'hypothèse de localité thématique (Davison, 2000).

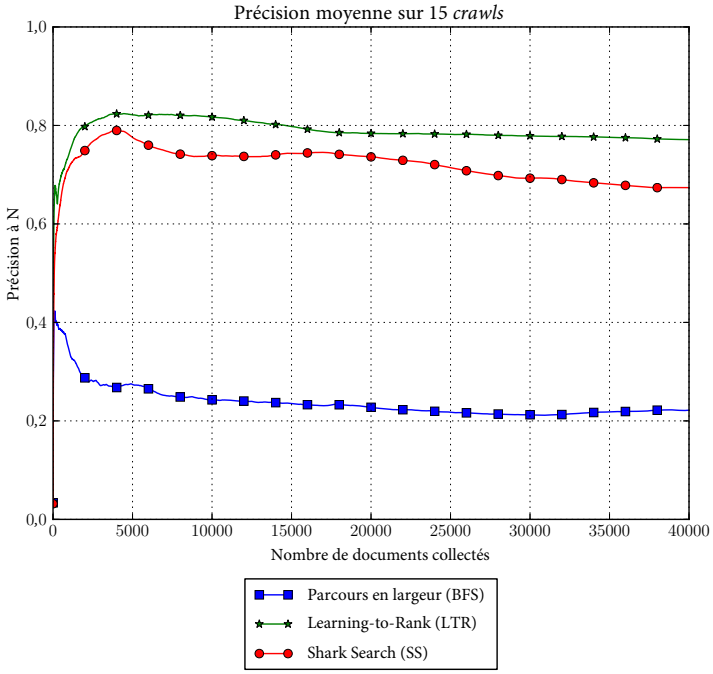

Figure 4. *Comparaison de la précision moyenne sur 15 thèmes des trois stratégies de crawling en fonction du nombre de documents collectés.*

Tableau 4. *Comparaison de la précision moyenne à N documents sur 15 thèmes.*

Précision à	10 000	20 000	30 000	40 000
Parcours en largeur	0,24 ($\pm$ 0,25)	0,23 ($\pm$ 0,24)	0,21 ($\pm$ 0,22)	0,22 ($\pm$ 0,23)
Shark search	0,74 ($\pm$ 0,20)	0,74 ($\pm$ 0,22)	0,69 ($\pm$ 0,25)	0,67 ($\pm$ 0,26)
Learning-to-Rank	**0,82** ($\pm$ 0,11)	**0,78** ($\pm$ 0,14)	**0,78** ($\pm$ 0,15)	**0,77** ($\pm$ 0,17)

L'algorithme Shark Search offre une précision moyenne bien plus haute que le parcours en largeur. Sa précision à 10 000 documents est de $0,74$ puis décroit jusqu'à $0,67$ à 40 000 documents. Le tableau 4 montre des résultats très variables en fonction des thèmes avec un écart type allant de $\pm 0,20$ à $\pm 0,26$. L'algorithme Learning-to-Rank surpasse l'algorithme Shark Search sur l'ensemble du *crawl*. À 10 000 documents, l'algorithme Learning-to-Rank surpasse l'algorithme de Shark Search de 8% en précision. Son gain maximal est à 40 000 documents où le gain augmente jusqu'à 10%. Le tableau 4 montre également une précision très stable pour cet algorithme, notamment après 20 000 documents (-0,01 entre 20 000 et 40 000 documents) alors

que l'algorithme Shark Search perd $0,07$ point sur ce même intervalle. De plus, nous observons que l'approche Learning-to-Rank obtient un écart type de $\pm 0,17$ sur l'ensemble des 15 thèmes ($\pm 0,26$ pour Shark Search), ce qui en fait une stratégie plus sure lorsqu'appliqué à de nouveaux thèmes. Enfin, nous insistons à nouveau sur le fait que l'algorithme Learning-to-Rank peut être amélioré facilement en intégrant de nouveaux traits dans la fonction d'ordonnancement, ce qui ouvre la porte à de nombreux travaux futurs.

5. Conclusions

Dans cet article, nous avons proposé d'apprendre une fonction d'ordonnancement pour ordonner la frontière de *crawl* d'un *crawler* orienté. Nous avons défini une approche auto supervisée pour générer des données de *crawl* annotées et apprendre une fonction d'ordonnancement indépendante du thème. Nous pensons que notre approche permet d'intégrer et de pondérer un grand nombre de traits de manière unifiée. Par ailleurs, nous avons tenté d'intégrer une notion de *tunnel* dans notre fonction d'ordonnancement qu'il nous faudra d'évaluer dans nos travaux futurs. Appliquée dans un cadre concret, notre stratégie a montré une précision supérieure à l'algorithme Shark Search, tout en offrant des performances plus stables sur l'ensemble des thèmes. Nous pensons que ces résultats sont prometteurs et que de nouveaux traits pourraient être ajoutés pour améliorer encore ces performances.

Nous avons identifié un certain nombre de pistes pour nos travaux futurs :

1) Tout d'abord, nous avons tenté d'intégrer l'effet tunnel dans notre modèle. Il serait intéressant de connaître l'effet de cette prise en compte, par exemple en entraînant un nouvel algorithme d'ordonnancement qui ne prendrait pas en compte cet effet (avec une échelle de valeurs binaire).

2) Nous avons fait l'hypothèse que nous pouvions définir une fonction d'ordonnancement indépendante du thème. Nous pensons que cette hypothèse est attractive mais simplificatrice, car notre modèle ne peut apprendre les relations entre thèmes qui semblent pourtant une information pertinente pour l'effet tunnel (Chakrabarti *et al.*, 2002 ; Diligenti *et al.*, 2000).

3) Nous avons proposé d'apprendre une fonction d'ordonnancement à partir de données de *crawl* générées spécifiquement pour notre tâche. Néanmoins, cette approche fournit un graphe de liens incomplet. Pour estimer avec plus de précision si une page est un tunnel, nous devrions nous appuyer sur un graphe de lien plus complet (en nous basant par exemple sur la collection ClueWeb09[7]). Cette approche serait toutefois difficile techniquement, car elle demanderait de traiter un très grand nombre de pages Web (millions, milliards).

4) Enfin, une dernière limite de notre approche est son incapacité à prendre en compte des traits dynamiques, c'est-à-dire extraits durant le *crawl* (PageRank, co-

7. `http://www.lemurproject.org/clueweb09.php/`

citation (Chakrabarti *et al.*, 2002)). Pour inclure ces traits, nous pourrions considérer les graphes de *crawl* d'entraînement à plusieurs moments du *crawl*.

6. Bibliographie

Aslam J., Kanoulas E., Pavlu V., Savev S., Yilmaz E., « Document selection methodologies for efficient and effective learning-to-rank », *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, p. 468-475, 2009.

Baroni M., Ueyama M., « Building general-and special-purpose corpora by Web crawling », *Proceedings of the 13th NIJL international symposium, language corpora : Their compilation and application*, p. 31-40, 2006.

Baykan E., Henzinger M., Marian L., Weber I., « Purely URL-based topic classification », *Proceedings of the 18th international conference on World Wide Web*, p. 1109-1110, 2009.

Bergmark D., Lagoze C., Sbityakov A., « Focused crawls, tunneling, and digital libraries », *Lecture notes in computer science*, 2002.

Castillo C., Effective web crawling, PhD thesis, University of Chile, June, 2005.

Chakrabarti S., den Berg M. V., Dom B., « Focused crawling : a new approach to topic-specific Web resource discovery », *Computer Networks*, vol. 31, n° 11-16, p. 1623-1640, 1999.

Chakrabarti S., Punera K., Subramanyam M., « Accelerated focused crawling through online relevance feedback », *Proceedings of the 11th international conference on World Wide Web*, p. 148-159, 2002.

Chapelle O., Chang Y., « Yahoo ! learning to rank challenge overview », *Journal of Machine Learning Research-Proceedings Track*, vol. 14, p. 1-24, 2011.

Chen J., Power R., Subramanian L., Ledlie J., « Design and implementation of contextual information portals », *Proceedings of the 20th international conference companion on World wide web*, p. 453-462, 2011.

Cho J., Garcia-Molina H., Page L., « Efficient crawling through URL ordering », *Computer Networks and ISDN Systems*, vol. 30, n° 1-7, p. 161-172, 1998.

Davison B. D., « Topical locality in the Web », *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, p. 272-279, 2000.

De Bra P., Post R., « Information retrieval in the World-Wide Web : making client-based searching feasible », *Computer Networks and ISDN Systems*, vol. 27, n° 2, p. 183-192, 1994.

Diligenti M., Coetzee F., Lawrence S., Giles C., Gori M., « Focused crawling using context graphs », *Proceedings of the 26th International Conference on Very Large Data Bases*, p. 527-534, 2000.

Duh K., Kirchhoff K., « Learning to rank with partially-labeled data », *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, p. 251-258, 2008.

Fan R., Chang K., Hsieh C., Wang X., Lin C., « LIBLINEAR : A library for large linear classification », *The Journal of Machine Learning Research*, vol. 9, p. 1871-1874, 2008.

Finn A., Kushmerick N., Smyth B., « Fact or fiction : Content classification for digital libraries », *DELOS Workshop : Personalisation and Recommender Systems in Digital Libraries*, 2001.

Friedman J. H., « Greedy function approximation : a gradient boosting machine », *Annals of Statistics*, vol. 29, n° 5, p. 1189-1232, 2001.

Herbrich R., Graepel T., Obermayer K., « Large margin rank boundaries for ordinal regression », *Advances in Neural Information Processing Systems*p. 115-132, 1999.

Hersovici M., Jacovi M., Maarek Y., Pelleg D., Shtalhaim M., Ur S., « The shark-search algorithm. An application : tailored Web site mapping », *Computer Networks and ISDN Systems*, vol. 30, n° 1-7, p. 317-326, 1998.

Li H., *Learning to Rank for Information Retrieval and Natural Language Processing*, vol. 12, Morgan & Claypool publishers, 2011.

Liu T.-Y., « Learning to rank for information retrieval », *Foundations and Trends in Information Retrieval*, vol. 3, n° 3, p. 225-331, 2009.

Liu T.-Y., Yang Y., Wan H., Zhou Q., Gao B., Zeng H.-J., Chen Z., Ma W.-Y., « An experimental study on large-scale web categorization », *Special interest tracks and posters of the 14th international conference on World Wide Web*, p. 1106-1107, 2005.

Lv Y., Moon T., Kolari P., Zheng Z., Wang X., Chang Y., « Learning to model relatedness for news recommendation », *Proceedings of the 20th international conference on World wide web*, p. 57-66, 2011.

Manning C., Raghavan P., Schutze H., *Introduction to information retrieval*, Cambridge University Press, 2008.

Menczer F., Pant G., Srinivasan P., « Topical web crawlers : Evaluating adaptive algorithms », *ACM Transactions on Internet Technology*, vol. 4, n° 4, p. 378-419, 2004.

Metzler D., Croft W. B., « Linear feature-based models for information retrieval », *Information Retrieval*, vol. 10, n° 3, p. 257-274, 2007.

Olston C., Najork M., « Web Crawling », *Foundations and Trends in Information Retrieval*, vol. 4, n° 3, p. 175-246, 2010.

Pang B., Lee L., « Seeing stars : Exploiting class relationships for sentiment categorization with respect to rating scales », *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, vol. 43, p. 115, 2005.

Pant G., Srinivasan P., « Link contexts in classifier-guided topical crawlers », *Knowledge and Data Engineering, IEEE Transactions on*, vol. 18, n° 1, p. 107-122, 2006.

Qin T., Liu T., Xu J., Li H., « LETOR : A benchmark collection for research on learning to rank for information retrieval », *Information Retrieval*, vol. 13, n° 4, p. 346-374, 2010.

Ramage D., Rafferty A., Manning C., « Random walks for text semantic similarity », *Proceedings of the 4th TextGraphs workshop*, ACL, p. 23, 2009.

Rennie J., McCallum A., « Efficient web spidering with reinforcement learning », *Proceedings of the 16th international conference on Machine Learning*, 1999.

Sculley D., « Large Scale Learning to Rank », *Proceedings of the NIPS Workshop on Advances in Ranking*, 2009.

Wu Q., Burges C. J., Svore K. M., Gao J., « Adapting boosting for information retrieval measures », *Information Retrieval*, vol. 13, n° 3, p. 254-270, 2010.