Scraping & Crawling

Collecter et exploiter les données du Web

Table des matières

Introduction

Depuis sa création au début des années 1990, le *World Wide Web*, communément appelé Web, a connu une croissance exponentielle. De quelques centaines de sites en 1993, sa taille estimée dépasse aujourd'hui les 45 milliards de pages[1]. Effet direct de cette croissance : les données qu'il contient sont progressivement devenues extrêmement intéressantes pour les entreprises. Un nouveau défi est alors apparu, il leur faut apprendre à recueillir, comprendre et exploiter les informations issues de cette source quasi inépuisable et sans cesse renouvelée. On observe, par exemple, la multiplication des comparateurs pour tous les types de produits (assurances, banques, produits culturels, etc.) qui centralisent les données collectées de différents sites. On peut également imaginer une enseigne de supermarché étudier la concurrence de façon automatique pour ajuster ses prix.

C'est dans le cadre de cette problématique que s'inscrivent les concepts de crawling et scraping, outils majeurs de la collecte de données sur le Web. Nous étudierons également le principe d'object search et le mouvement émergent de l'open data. Par la suite, nous mettrons en relief le scraping par le biais d'un cas d'utilisation. Enfin nous ferons le point sur la juridiction qui entoure ces pratiques.

1. Définitions - Théorie - Principes - Algorithmes

Nous entamons notre synthèse en étudiant les différentes techniques existantes pour explorer le Web. Tout d'abord le crawling ayant pour objectif de comprendre l'organisation du Web puis le scraping qui s'attarde sur la récupération de données dans un environnement stable. L'object search, une pratique plus récente qui fait le lien avec le Web sémantique où l'information est structurée. Enfin nous parlerons de l'open data, un mouvement émergent qui cherche à faciliter la mise à disposition de données.

1.1. Crawling

Originellement, le crawling consiste à parcourir et indexer le Web afin d'en établir la cartographie. Le programme qui réalise cette tâche de façon automatique est appelé Web crawler. Suivant les époques et les ouvrages utilisés pour notre étude, il pourra prendre différentes appellations : web spider, webbot...

Le besoin de cartographie et d'indexation ayant émergé en même temps que la création du Web lui-même, le premier Web crawler, the World Wide Web Wanderer, a été publié au printemps 1993. Il a permis pendant 3 ans de suivre la taille du Web afin de mesurer sa croissance.

[1] http://www.worldwidewebsize.com

1.1.1. Fonctionnement

Pour comprendre comment fonctionne un crawler, il est nécessaire de rappeler le fonctionnement du Web lui même. Celui-ci peut être considéré comme un ensemble de ressources, chacune identifiée par une adresse unique appelée URL[2] ou adresse Web. Dans le cas où la ressource considérée est une page web, celle-ci pourra être affichée dans un navigateur web. Chaque page web est susceptible de contenir un certain nombre de liens vers d'autres ressources, celles-ci pouvant être d'autres pages web.

De façon conceptuelle, l'algorithme permettant d'implémenter un crawler est alors très simple. Dans sa version la plus élémentaire, celui-ci n'utilise qu'une seule structure de données : un ensemble d'adresses web connues. La structure ne contient au départ qu'un petit nombre d'adresses web appelées candidats initiaux. Pour chacune de ces adresses, le programme télécharge la page web associée et identifie les liens qu'elle contient. Pour chacun de ces liens, si l'adresse est inconnue, elle est ajoutée à la liste. L'algorithme choisit alors un nouvel ensemble d'adresses à visiter parmi celles qu'il vient de découvrir et répète le processus.

1.1.2. Limites

Le fonctionnement du web étant presque le même depuis sa création, les limites rencontrées par les premières implémentations de crawler [1] sont les mêmes que celles auxquelles peuvent être confrontées les entreprises actuelles [2]. Elles se déclinent selon différents aspects.

1.1.2.1. Performances

Le premier de ces problèmes est celui de la performance. Les premières versions des crawlers ont été confrontées à la croissance exponentielle du Web tandis que de nos jours, c'est plus le volume de données et la fréquence de leur changement qui est mis en cause. Les problèmes de performance se présentent sous trois aspects principaux d'après un article sur les meilleurs stratégies de crawling [3].

Tout d'abord, c'est la bande passante et l'espace disque qui limitent la quantité de pages crawlées car ils sont ni infinis, ni gratuits. On estime aujourd'hui à $8x10^9$ pages indexées par Google. Avec une moyenne de 15kB par page, il faudrait 4 millions de dollars pour récupérer et stocker tout le Web. Pour maintenir une base relativement à jour, un crawler doit donc télécharger chaque seconde un nombre de pages pouvant atteindre plusieurs milliers. Au delà de la contrainte de la politesse qui impose de ne pas surcharger les sites que l'on est en train de parcourir, il faut s'assurer que l'infrastructure réseau utilisée par la machine qui héberge le crawler soit suffisante.

Ensuite, les pages sont en perpétuel changement. Le crawling est une image à un instant T mais lorsqu'une recherche est effectuée, on peut trouver des liens morts et des pages inexistantes. Le crawler doit donc par moment arrêter de télécharger des nouvelles

[2] Uniform Resource Locator

ressources et vérifier voire mettre à jour les pages déjà indexées.

Enfin, en considérant qu'une page est définie par son URL, alors le Web est infini surtout avec l'apparition de pages dynamiques. Afin de conserver de bonnes performances, un crawler doit donc mettre en place une stratégie pour télécharger les pages les plus "*importantes*" en premier lieu et ne pas oublier de mettre à jour la base de données.

1.1.2.2. Politesse et bonnes pratiques

Vient ensuite la question de la politesse et des bonnes pratiques du Web. Le but d'un crawler n'est bien évidemment pas de nuire aux sites visités. Cependant, plusieurs facteurs peuvent avoir des effets négatifs sur celui-ci. Dans le cas d'un site comportant un grand nombre de pages, après quelques itérations du crawler, la liste des pages à télécharger peut atteindre plusieurs milliers. Si aucune précaution n'est prise par le crawler et que celui commence le téléchargement simultané de plusieurs pages, le site ciblé est susceptible de subir des ralentissements ou même d'être rendu indisponible. Les mesures utilisées généralement consistent à limiter la bande passante utilisée lors du crawl d'un site ou à ne télécharger qu'une seule page à la fois.

1.1.2.3. Protection

Pour formaliser ces pratiques, il existe plusieurs façons de limiter voir bloquer l'action des crawlers [4]. Certaines entreprises comme Impreva [5] commercialisent même des solutions mettant en oeuvre ces techniques de protection. Celles-ci peuvent prendre divers aspects, à commencer par le plus simple : le fichier "robots.txt" ou la balise HTML "meta robots". Ils permettent de spécifier de façon normalisée quelles parties du site un crawler est autorisé à visiter ou non.

L'algorithme utilisé par un crawler respectant la convention du fichier robots.txt reste relativement simple. Avant de télécharger la première page d'un site qui n'a jamais été visité, le crawler télécharge le fichier robots.txt depuis le serveur. Le fichier contient une liste d'adresses interdites. Pour chaque adresse que le crawler découvrira sur ce site, il lui suffira de vérifier qu'elle n'est pas dans la liste de celles qui sont interdites avant de télécharger la ressource associée.

Un site pourra par exemple demander aux crawlers de ne pas visiter des pages qui demandent beaucoup de ressources pour être générées ou qui n'ont aucun intérêt. Ces deux mesures relèvent cependant uniquement de la bonne volonté du crawler, qui pourra toujours décider de les ignorer et parcourir l'intégralité du site. Bien entendu, la majorité des crawlers commerciaux qui parcourent le Web (ex : les moteurs de recherche) respectent ces instructions, protégeant effectivement les sites de la surcharge qui aurait pu être occasionnée par le rendu des pages en question.

Il est également possible pour les sites web de mettre en place des méthodes de filtrage basées sur le "user agent". Toute application qui se connecte à un site web fournit cet identifiant au site sous forme d'une chaîne de caractères. Il est propre à chaque navigateur web, client en ligne de commande, crawler, etc. Un site pourra donc restreindre l'accès à

certaines ressources en fonction du user agent fourni lors de la connexion. Ceci ne pouvant se faire de manière immédiate comme avec l'utilisation du fichier robots.txt, l'intérêt des solutions que peuvent proposer des entreprises comme Impreva commence à se préciser. Cependant, rien n'empêche un crawler de fournir un user agent de navigateur web afin de se faire passer pour un utilisateur réel.

D'autre méthodes plus avancées existent pour contrer les crawlers. On pourra par exemple détecter de trop nombreuses connexions depuis la même machine et bannir son adresse IP pour empêcher toutes tentatives de crawl ultérieures. Authentifier les utilisateurs pour accéder à certaines parties du site est également une bonne méthode pour s'assurer qu'elles ne seront pas visitées par un robot. L'utilisation du javascript pour générer certaines parties des pages web pourra empêcher un crawler d'y avoir accès et sera complètement transparent pour un utilisateur réel. De telles méthodes seront plus utilisées pour empêcher la collecte d'information par du scraping, que nous verrons plus en détail par la suite, plutôt que pour empêcher le crawling.

1.1.3. Architecture physique

Le Web ayant connu une croissance exponentielle à ses débuts, les implémentations simplistes se sont très vites retrouvées insuffisantes. Pour augmenter les capacités des crawlers, il a donc été nécessaire de réfléchir à de nouvelles implémentations permettant d'utiliser plusieurs machines afin de répartir la charge de travail. Typiquement, le stockage des données récoltées par les crawlers sera répartie entre plusieurs machine qui pourront se situer dans des lieux géographiques différents. Par exemple, les données issues du crawling de sites web de chaque continent seront stockées dans des datacenter locaux. De la même façon, les machines effectuant le crawling auront tendance à être placées de façon optimisée par rapport aux sites ciblés.

Mettre en place une architecture décentralisée impose cependant de répondre à de nouvelles contraintes. Il faut s'assurer que les sites visités par les différentes machines exécutant le crawler sont bien disjoints. Ceci pourra se faire en partitionnant l'espace des adresses web et en assignant chaque partie à une instance différente du crawler.

Ce type d'architecture physique peut être modélisé par le schéma suivant [6] :

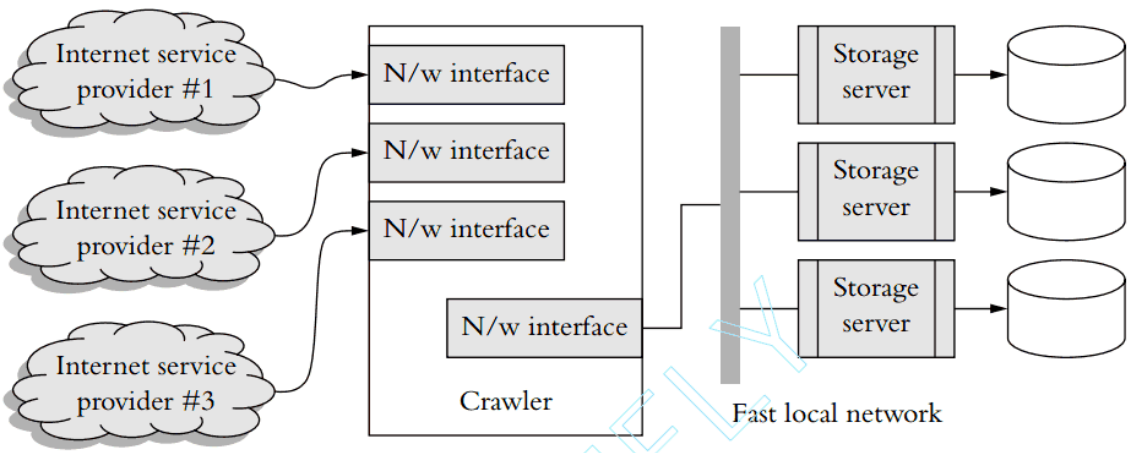

Figure 1 : architecture de crawler distribué

Dans ce cas, le crawler en lui-même n'utilise qu'une seule machine mais plusieurs interfaces réseau afin de bénéficier d'une plus grande capacité de téléchargement. Le stockage des informations récoltées est effectué sur plusieurs machines différentes accessibles en réseau local. De cette façon, les temps d'écriture des données sont moins limitants. Cette architecture très similaire à celle proposée dans l'article de Marc Najork [1]. En effet, celui-ci décrit également une architecture dans laquelle le processus de téléchargement est distribué et les données réparties entre plusieurs serveurs. Cependant, dans son cas, le crawler lui-même est réparti entre plusieurs machines, ce qui n'est pas le cas ici. La solution qu'il propose semble donc encore plus évolutive et flexible que celle présentée en figure 1.

1.1.4. Architecture logicielle

On peut modéliser les différents composants d'un crawler de la façon suivante [7] :

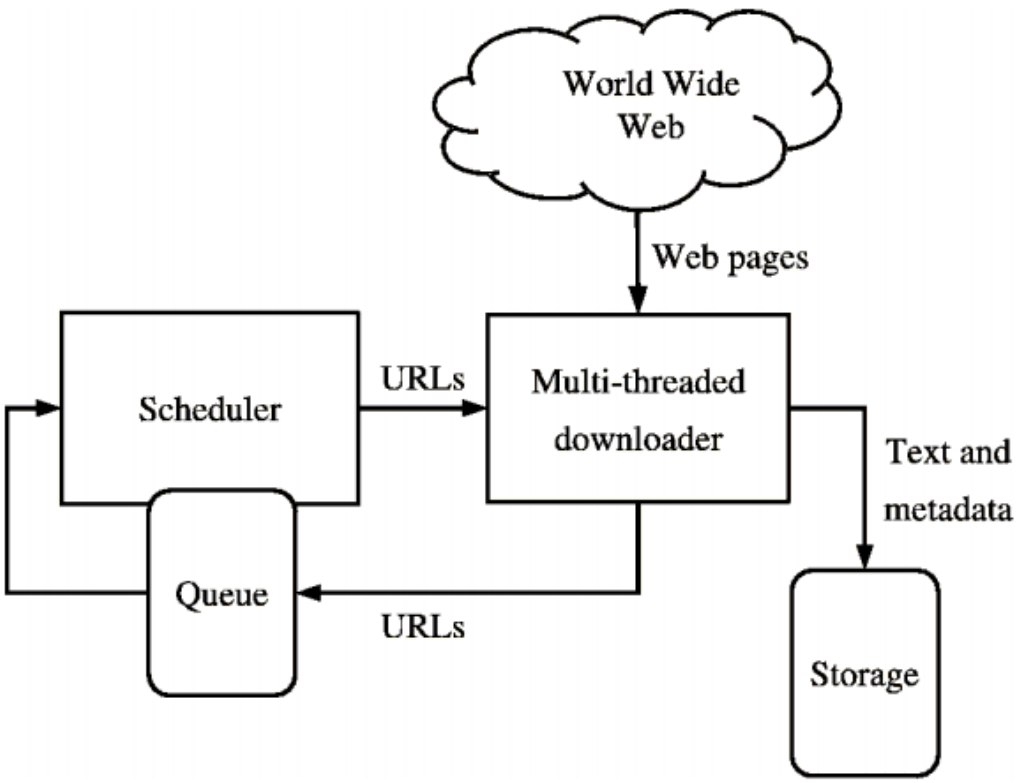

Figure 2 : composants logiciels d'un crawler

Cette représentation assez haut niveau représente la base requise pour créer un crawler extensible et efficace. L'utilisation d'un ordonnanceur (scheduler) pour prioriser les URLs à visiter permet de mieux gérer les différents temps morts entre le téléchargement des pages.

Soumen Chakrabarti [6] propose par exemple de grouper les pages d'un même site web et d'utiliser le temps nécessaire à leur téléchargement pour émettre les requêtes de résolution DNS des URLs qui suivent dans la file. Ainsi, celles-ci seront prêtes à être visitées quand leur

tour viendra et le téléchargement pourra commencer directement.

1.1.5. Utilisations

Comme précisé en introduction, le but premier des crawlers était de mesurer et cartographier le Web. Le contenu des pages téléchargées par les crawlers n'était alors utilisé que pour découvrir de nouvelles URLs. Rapidement, les exemples de programmes utilisant les données issues de crawlers sont multipliés. Si le fonctionnement de base restait le même, le contenu des pages était cette-fois analysé, notamment à des fins d'indexation : c'est la naissance des moteurs de recherche.

Le nombre de pages téléchargées par un crawler pouvant atteindre de très grands nombres, une fois que des données ou meta-données en sont extraites, il est très souvent intéressant d'analyser ces dernières. Entrent alors en jeu des techniques de data mining permettant d'en extraire des informations. Ces information peuvent ensuite permettre de créer d'innombrables services : comparateur de prix, aggrégateur de données, annuaires…

L'analyse des données téléchargées par le crawler peut également aider celui-ci à mieux prioriser les URLs à visiter. Il est en effet possible de créer différents indicateurs de valeur potentielle d'une URL à partir des données de la page qui la contient ou les autres pages d'un même site. En procédant de la sorte, le crawler pourra choisir de laisser de côté certaines URLs et d'en visiter certaines autres en priorité. Ainsi, la valeur apportée par le programme est supérieure à celle qu'il aurait dans le cas d'une approche plus naïve. Cette façon de procéder est appelée "focused crawling".

L'article de recherche de S. Chakrabarti [8] explique la méthode utilisée pour choisir les liens à suivre. La solution contient trois composants. Un *classifier* qui détermine la catégorie à laquelle appartient une page en la comparant à un ensemble de page dont les catégories sont déjà connues. En réalité, une page appartient souvent à plusieurs catégories mais leur solution est une première approche. Un *distiller* qui défini la valeur d'une page en fonction de sa catégorie et les liens qu'elle contient. Enfin le *crawler* qui interroge le *classifier* et le *distiller* pour choisir les liens à suivre. Ainsi un ensemble de catégories est choisi au début du crawling et les pages téléchargées sont en principe toutes en rapport avec les sujets recherchés.

1.2. Scraping

1.2.1. Fonctionnement

Le principe de base du scraping est simple : transformer des informations non structurées présentes dans des pages web en données structurées facilement exploitables. Le programme qui met ce concept en oeuvre est généralement appelé "scraper".

Titre	Auteur	Date	Thème
Ingénieurs « tous les clignotants sont au vert »	Benoît Floc'h	10.12.2013	Éducation
Yahoo!, l'un des rares survivants des débuts du Web, fête ses 20 ans	Sarah Bellouezanne	03.01.2014	Technologies

Figure 3 : Exemple de transformation de données, lemonde.fr vers une BDD

Contrairement à un crawler qui découvre lui-même les sites parcourus et les pages web téléchargées, le scraper travaille sur un site ou ensemble de sites connus par avance. Le scraper pourra alors être un logiciel commercial paramétré de façon à récupérer les données souhaitées sur le site en question ou bien un programme développé spécifiquement pour cette tâche et donc parfaitement adapté au site.

Suivant le type, la quantité d'informations à récupérer ainsi que la complexité pour y accéder, les architectures logicielles et matérielles nécessaires à la mise en place d'un scraper varient d'un extrême à l'autre. Ainsi les démonstrations effectuées par Andrew Peterson [9] et Charles Severance [10] nécessitent uniquement un ordinateur personnel et le code nécessaire à l'implémentation de leurs scrapers est très limité. Dans les deux cas, l'information récupérée provient d'un seul site web et est utilisable de façon quasi immédiate.

Inversement, l'exemple [2] proposé par Data Publica à travers le travail réalisé pour la

société OuestMarchés demande des moyens considérables. Les données brutes sont collectées depuis 35 sources différentes, dans des formats très hétérogènes. Il leur a donc été nécessaire de mettre en place une architecture logicielle complète permettant de télécharger, nettoyer, convertir, stocker et visualiser les données cibles.

1.2.2. Mise en oeuvre

Comme nous venons de le voir, les objectifs d'un scraper peuvent varier d'un extrême à l'autre. En conséquence, les moyens employés pour la mise en oeuvre sont très variés. On distingue cependant deux grandes catégories : les logiciels prêts à l'emploi et les librairies permettant de développer des scrapers. Dans chacune de ces catégories, le panel de solution est également très large.

1.2.2.1. Logiciels prêts à l'emploi

Les premiers logiciels de cette catégorie sont les navigateurs internet. En effet, en visualisant la page et avec de simples copier/coller, un utilisateur peut récupérer les données d'une ou plusieurs pages afin de les réutiliser dans un autre contexte, ce qui correspond à la définition du scraping. Cette méthode étant bien entendu très fastidieuse et peu efficace, il existe des addons pour navigateurs permettant de faciliter ce processus (ex : DownThemAll[3] pour Firefox). Viennent ensuite toutes les solutions logicielles, professionnelles ou non, qui une fois paramétrées correctement peuvent se charger de toutes les étapes d'acquisition des données structurées.

1.2.2.2. Librairies

La plupart des langages de programmation disposent au moins d'une librairie basique permettant de travailler avec le web. La bibliographie étudiée comporte un certain nombre d'exemples d'utilisation de ces librairies dans plusieurs d'entre eux : PHP [4], java [6] ou encore Python [11]. En plus de ces librairies basiques permettant uniquement de télécharger des pages web, il existe de nombreux exemples permettant d'implémenter rapidement le reste du travail d'un scraper : parsing des pages, exécution du javascript, gestion de l'authentification…

1.2.3. Limites

Le support de travail des scrapers étant le même que les crawlers, les pages web, on retrouve la plupart des limites qui s'appliquaient à ces derniers. Le scraping étant globalement plus contraignant que le crawling (nous rappelons qu'il est nécessaire de connaître par avance la structure du site ciblé), il existe des mesures spécifiques pour l'empêcher.

Certains sites changent régulièrement la structure du code html de leurs pages. S'il est possible que cela ne change rien visuellement dans un navigateur web, cela va empêcher le programme d'en extraire les données. Bien entendu, le créateur du scraper pourra toujours le corriger pour prendre en compte les modifications mais il ne sera pas à l'abri d'autres changements ultérieurs. Ce type de mesure ne change en revanche absolument rien au

[3] https://addons.mozilla.org/en-US/firefox/addon/downthemall/

fonctionnement d'un crawler qui se contente de détecter les liens présents dans une page, quelle que soit la structure html de celle-ci.

1.2.4. Architecture physique

Contrairement aux crawlers pour lesquels le domaine exploré est virtuellement illimité, les scrapers sont destinés à agir sur un nombre réduit et connu par avance de sites web. S'il sera possible de rencontrer des crawlers utilisant un parc de plusieurs milliers de machines, un scraper n'aura en général besoin que d'une seule machine, en tout cas pour la partie chargée du téléchargement des données.

1.2.5. Architecture logicielle

Le scraping étant très spécifique aux sites et données cibles, il est difficile de donner une architecture applicative type. On retrouvera cependant toujours certains composants, parfois similaires à ceux identifiés dans les crawlers. La finalité d'un scraper étant de télécharger des ressources, on retrouvera donc certainement une file d'URLs en attente de téléchargement. Pour de meilleures performances, ils auront aussi tendance à utiliser des téléchargements simultanés grâce au multithreading, exactement à la façon des crawlers.

Autre élément fortement similaire aux crawlers : un scraper devra stocker les informations recueillies et disposera donc d'un module permettant la persistance, que celle-ci se fasse sur disque, en base de données, ou en réseau.

1.3. Object search

1.3.1. Définition

La recherche par objet s'oppose bien souvent à la recherche par page traditionnelle dont Google est l'exemple le plus connu et pour laquelle l'utilisateur se voit fournir une liste de pages qu'il devra par la suite explorer afin d'en retirer les informations souhaitées. Dans cette nouvelle approche, l'information est stockée de manière plus structurée autour d'une entité appelée *objet*. Cet objet peut faire référence à une personne, un film, etc.

De manière similaire à la programmation objet, chacune de ces entités possèdera un certain nombre d'attributs qui seront porteurs de l'information connue sur cet objet. Un film possèdera par exemple un attribut *titre*, un autre *durée*, ainsi qu'un grand nombre d'autres, dont certains sont eux mêmes des objets (comme dans cet exemple le *réalisateur* du film, qui possède lui aussi des attributs).

Cette politique de stockage de l'information permet ensuite d'effectuer des recherches non pas sur des pages contenant des informations, mais directement sur les attributs présents dans les objets. Un exemple : la recherche "*Tous les films de George Lucas entre 1980 et 1995*" permettra de renvoyer différentes entités de films et d'obtenir les informations associées.

Si cette recherche par objet n'est pas encore très développée, des exemples existent déjà.

C'est par exemple le cas de Google, qui en plus de sa recherche par page propose maintenant une fiche technique de certaines entités comme les pays, les films ou les personnes célèbres.

La recherche par objet a trait au courant actuel de web sémantique, qui se focalise sur la liaison des différentes données présentes sur internet afin d'en simplifier l'accès. En effet, dans l'esprit du web sémantique, la recherche par objet vise à structurer l'information afin d'augmenter le potentiel de ces connaissances.

1.3.2. Avantages

Il existe plusieurs avantages notables à ce type de recherche :

- **Gain de temps lors de l'extraction des informations.** L'avantage le plus évident de la recherche par objet est certainement qu'il permet d'économiser un temps parfois considérable lors de la recherche d'information en extrayant directement les informations d'une page parfois complexe et en les filtrant conformément à la requête formulée par son auteur [12]. Si ce gain de temps peut se révéler négligeable sur des recherches triviales, comme l'année de publication d'un livre, il peut être en revanche bien plus important lors d'interrogations plus complexes qui nécessiteraient une phase de recoupage plus longue, comme par exemple *"Le rôle féminin qui a le plus joué avec Brad Pitt".*
- **Recoupage d'informations préalable.** On le sait, internet n'est pas toujours une source d'information très fiable. Pour pouvoir récupérer des informations plus dignes de confiance, il est d'usage de recouper plusieurs sources d'information afin de les confronter. Le problème est à la fois le temps nécessaire à cette démarche et le risque de recouper plusieurs informations provenant de la même source, ce qui en annule l'intérêt. En se basant sur plusieurs sources d'information et en évaluant leur qualité relative, les algorithmes d'obtention d'objet permettent encore une fois un gain de temps important [8].
- **Meilleure mise à disposition des informations pour les API.** Si les utilisateurs humains peuvent quand même en général retrouver les informations via des moteurs de recherche traditionnels, ce n'est pas forcément le cas des différents logiciels qui nécessitent des informations des moteurs de recherche, obligeant par exemple le recours à des crawlers ou à des scrappers. La mise à disposition d'informations structurées permettrait un accès bien plus facile à ces dernières.

1.3.3. Limites

Bien qu'il présente un nombre important d'avantages, des limites de cette recherche par objet existent, notamment au niveau de la mise en place de la base d'objets. Les principales difficultés sont :

- **Choix des objets et des structures correspondantes.** Le principal problème, lorsqu'on tente d'appréhender les objets du réel via l'informatique, consiste à réussir à inculquer à la machine la notion de ce qu'est cet objet [13] (par exemple, qu'un

roman est représenté par son titre, son auteur, et tout un ensemble d'autres attributs). Pour l'heure, ceci ne peut se faire que par intervention humaine avec les limites que cela implique, notamment les risques d'erreurs et d'oublis [12]. Ceux-ci peuvent avoir des conséquences très néfastes sur l'utilité d'une base de données constituée par crawling. En effet, si l'on se rend compte après coup qu'un attribut essentiel a été oublié lors de la spécification initiale, il sera nécessaire de reprendre du début la récupération d'information pour ce type d'entité afin de compléter l'ensemble des champs manquants.

- **Identification des entités.** Une fois les structures des objets définies, le remplissage de la base reste problématique [13]. Il est en effet nécessaire, avant de définir les attributs d'un objet, d'identifier une entité et de lui attribuer le bon type, ce qui permettra par la suite de remplir les valeurs des attributs. Pour définir le type d'entité, les algorithmes se basent le plus souvent sur plusieurs éléments qu'ils recouperont :
    - La désignation : un nom se trouvant après le mot "Auteur :" en sera probablement un.
    - Les attributs présents : une entité sur la page de laquelle un réalisateur, des acteurs et un titre est présent sera probablement un film par exemple.
    - L'emplacement : une entité dont la page présente la même structure (sur le même site) qu'une autre décrivant un livre sera certainement un livre également.
- **Recoupage des différents articles.** Comme présenté précédemment, le manque de fiabilité de certaines ressources présentes sur internet entraîne le besoin d'un recours à une *cross-validation* pour obtenir une information relativement fiable. Pour cela, les algorithmes actuels font appel à un système de notation des différentes sources d'information (i.e. des sites).

1.3.4. Futur de l'object search

L'approche de la recherche par objet est devenue populaire ces dernières années et se démocratisera probablement dans un futur proche. Les géants du web proposent déjà des applications du web de l'objet telles la "*Recherche dans le graphe*" de Facebook ou les infoboxes de Wikipedia. Si la mise en place de telles pratiques présente énormément d'intérêt, il reste encore de nombreux défis à relever pour qu'elle puisse se démocratiser. Un des défis des prochaines années sera le passage du web de l'objet au niveau international en regroupant les données provenant de l'ensemble des langues disponibles.

1.4. Open data

1.4.1. Présentation de l'open data

Une donnée ouverte (ou open data) est une donnée numérique d'origine publique ou privée. Elle est produite par une collectivité ou un service public (éventuellement délégué) et est diffusée de manière structurée selon une méthodologie et une licence ouverte garantissant son libre accès et sa réutilisation par tous, sans restriction technique, juridique ou financière. Le courant de l'open data remonte déjà à plus de 50 ans [14], ayant été

introduit dans le secteur de la recherche scientifique lors de l'International Geophysical ICSU (International Council of Science) en 1957.

Depuis, ce courant a été largement rejoint par de nombreux organismes (tant publics que privés) dans ce qui semble être une tendance qui se généralise de plus en plus. Les motivations poussant les différentes structures à rejoindre ce courant sont diverses : pour les entreprises privées, cette volonté de mettre en commun leurs données est bien entendu intéressée, cette opération permettant souvent soit un gain de notoriété et / ou visibilité, soit une plateforme d'accès à des partenaires permettant la création de services dédiés. La volonté des organismes publics rejoint quant à elle d'avantage des objectifs d'utilité publics, en permettant par exemple la création d'application mobiles en lien avec la SNCF ou la RATP [15] pour faciliter la vie des usagers.

Cette mise à disposition des données et / ou technologies peut aussi avoir pour objectif l'émergence d'un nouveau marché, l'exemple le plus frappant étant la divulgation par le gouvernement américain de la technologie (à l'origine militaire) du GPS, qui contribue actuellement à hauteur de 96 milliards de dollars par an à son économie [14] pour un coût de divulgation bien inférieur. Cette valeur économique peut également intervenir au niveau de la prévention de coûts :

"It has been estimated that, by unlocking the potential of big data, developed European economies could save between €150 billion and €300 billion annually in the form of operational efficiency gains and increased potential versus actual collection of tax revenue alone. A study has estimated the value of direct and indirect economic impacts of government-owned data across the EU-27 at €140 billion annually. It also estimated that, with lower barriers and improved infrastructure, this value could have been around €200 billion in 2008, representing 1.7% of the European GDP for that year." [14]

1.4.2. Développement

Bien qu'une des activités qui bénificierait le plus d'une diffusion massive des données sous forme d'open data soit le domaine de la recherche scientifique, la pratique montre que l'effort est plutôt fourni par les collectivités publiques et les états [14]. La plupart des gouvernements ont en effet publié des données sous forme de site open data, le premier exemple étant data.gov aux Etats-Unis en 2009 (l'équivalent français étant data.gouv.fr). La volonté d'ouverture de leurs données des entreprises publiques est en revanche très hétérogène et dépend notamment du mode de fonctionnement de ces dernières : certaines entreprises ont un historique de mise à disposition payante de leurs données pouvant représenter jusqu'à 20% de leur chiffre [14], l'ouverture et la mise à disposition gratuite de celles-ci serait un grave coup porté à leur économie.

La lenteur d'adaptation de l'ensemble des détenteurs de données à cette tendance s'explique par plusieurs raisons. Outre les implications économiques, l'open data pose surtout le problème des standards dans le traitement de données souvent stockées de manière disparâtre et sans lien entre elles [15], rendant techniquement difficile leur regroupement. D'autres aspects juridiques (développés plus loin dans la partie correspondante) concernant

l'obtention et la conservation des données privées viennent également entraver cette avancée.

Une autre question qui peut être soulevée est l'hétérogénéité des sources d'information pouvant marginaliser certains aspects. Un exemple simple : la recherche "*Cluc sport*" sur le site data.gouv.fr procure un nombre impressionnant de résultant s'apparant à des statistiques des clubs de sport de Digne-les-Bains. La masse d'information élevée transmise par cette commune est sans commune mesure avec la taille de celle-ci, ce qui est une constante dans ce domaine.

Un bon exemple de la diffusion des données ouvertes peut être montré par la politique menée dans le monde de la recherche. La recherche a en effet un besoin cruxial d'accès aux données, tant pour la problématique de vérification d'erreurs ou de fraudes dans les articles publiés que pour l'accès de tous à l'état de l'art afin de s'en servir comme base pour des travaux ou de les enrichir [14].

1.4.3. Outils techniques et API

Si l'accès par un humain des différentes informations disponibles est intéressant, le potentiel des données collectées est surtout révélé lors de son utilisation par des applications annexes permettant de fournir un service basé sur ces informations. De nombreuses applications existent, parmi lesquelles OpenStreetMap, un exemple particulièrement intéressant permettant d'utiliser les données mises à disposition pour générer une carte annotée des villes françaises. Cependant, à cause du nombre restreint de villes ayant fourni les données requises (Lyon étant cependant l'une d'entre elles), cette carte est grandement incomplète.

Il y a en effet un grand problème au niveau de la mise à disposition des données : de nombreux sites ne disposent pas d'API permettant aux programmes d'accéder aux données. C'est notamment le cas de data.gouv.fr, qui dans sa version actuelle ne dispose pas d'API utilisable (bien que celle-ci soit prévue dans un futur proche). Selon le blog RegardsCitoyens.org, ce portail reste en effet assez fermé, notamment à cause de nombreux formats de fichiers propriétaires (Microsoft). On voit donc bien qu'un travail important reste encore à fournir pour de nombreuses structures d'open data.

Il existe cependant des moyens détournés d'accéder à ces informations de manière automatique. Le soft (payant) d'OpenDataSoft permet par exemple d'accéder aux données fournies via des API Rest / Json. Le problème est bien évidemment que ce service devrait être inclus de base dans le service fourni par le gouvernement et non pas payant, ce qui va à l'inverse de la tendance open data.

Tout n'est cependant pas noir et de nombreuses bases sont pourvues d'API. Ces dernières sont en général présentes sous la forme de web services (SOAP ou REST) proposant des données sous forme majoritairement XML ou JSON.

2. Cas d'utilisation et techniques

2.1. Cas d'utilisation : scraper de replay Starcraft 2

2.1.1. Contexte

Starcraft 2 est un jeu vidéo populaire permettant à des joueurs de s'affronter via internet. Après une partie, un fichier contenant son déroulement peut être créé afin de garder une archive de la rencontre. Ces fichiers sont appelés "replays". Plusieurs sites d'actualité dédiés publient régulièrement ces fichiers.

Le but du programme est de télécharger les replays publiés par un ensemble de sites et d'en extraire le contenu significatif à des fins de data mining. Nous détaillerons toutes les étapes mises en oeuvre pour implémenter cette application. Ce cas d'utilisation est une partie de notre projet spécifique.

2.1.2. Architecture applicative

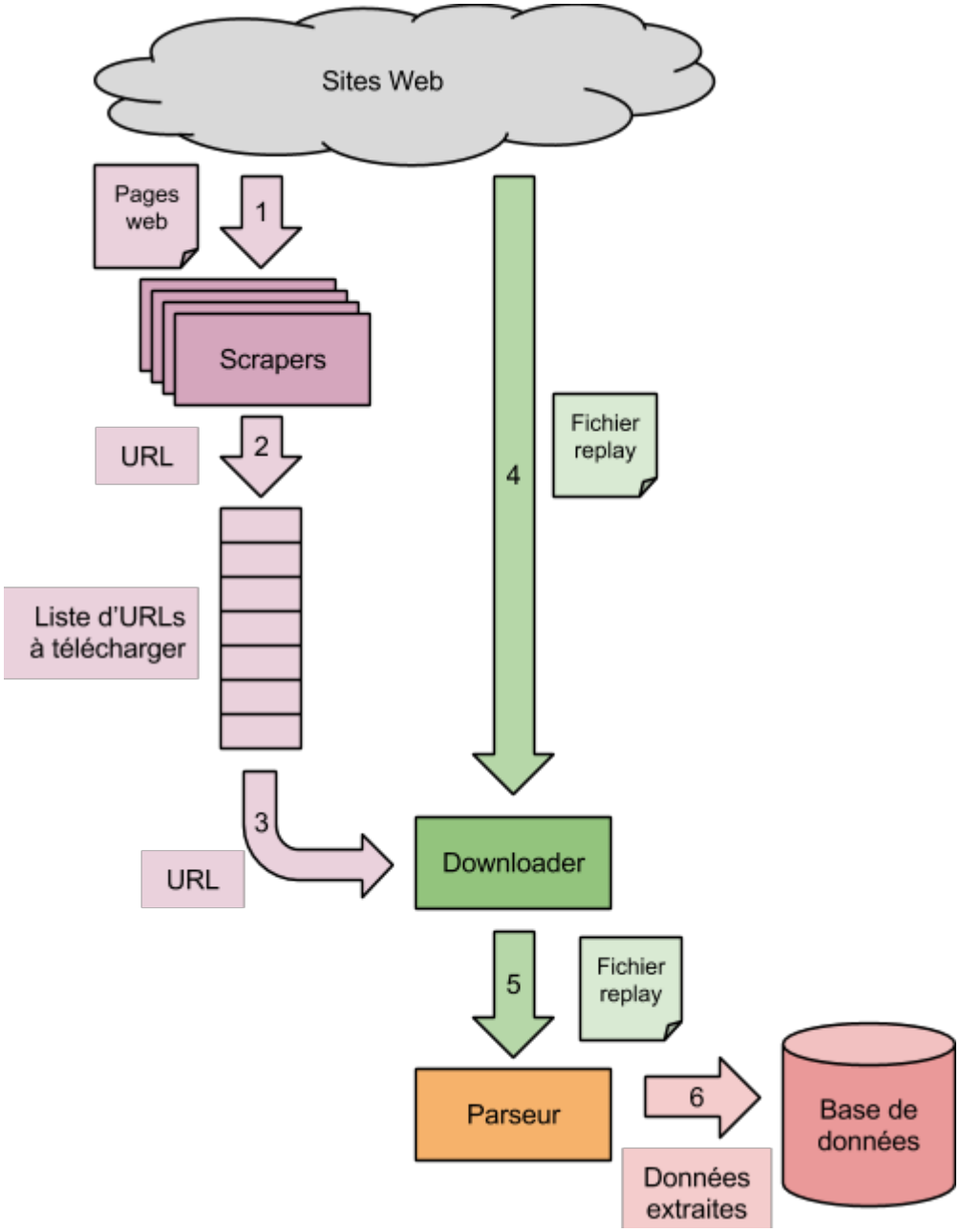

Figure 4 : architecture applicative

2.1.3. Scraping

L'ensemble des sites cibles étant connu, nous avons pu déterminer pour chacun d'entre eux un moyen d'accéder aux replays publiés. Nous avons ensuite implémenté un scraper pour chaque site. L'information fournie par un scraper est la suivante : la liste des URLs des nouveaux replays publiés sur le site depuis la dernière visite du scraper. Le reste du programme se charge alors de traiter cette liste. Pour extraire les liens des pages web, nous avons utilisé la librairie pyQuery[4] qui permet de manipuler de façon simple le code HTML d'une page et ainsi faciliter notre travail. Dans la plupart des cas, la liste des URLs des replays

[4] http://pythonhosted.org/pyquery/

proposés au téléchargement sera accessible via un simple sélecteur CSS.

Prenons par exemple le site http://sc2rep.com. La page qui contient les derniers replays publiés est la suivante : http://sc2rep.com/replays/index.

Figure 5 : derniers replays publiés

Comme on peut le voir sur cette capture d'écran, les replays sont classés par ordre de publication. On en déduit alors immédiatement l'algorithme permettant de détecter de nouveaux replays :

```
Télécharger la page http://sc2rep.com/replays/index
Interpréter le code HTML avec pyQuery
Extraire l'url du premier replay
Tant que l'url du replay est inconnue, faire :
     Ajouter l'url à la liste de téléchargement
     Extraire l'url du replay suivant
```

En exécutant cet algorithme périodiquement, tous les replays publiés sur ce site seront ajoutés à la liste de téléchargement.

2.1.4. Téléchargement et traitement

À partir de la liste d'URLs fournies par les différents scrapers, une autre partie du logiciel se charge de télécharger les fichiers de replay. Ceci se fait simplement en utilisant le module python dédié : urllib [10]. Notons que chaque scraper s'assure qu'il ne fournit aucune URL en double. En revanche, il est possible que deux scrapers fournissent un lien vers le même fichier replay sous deux noms différents.

Une fois le fichier téléchargé, les informations que nous cherchons en sont extraites par le parseur. C'est à ce moment que nous nous assurons que le replay n'est pas déjà connu. Les informations sont ensuite persistées dans une base de données pour un usage ultérieur par d'autres applications.

2.1.5. Architecture physique

Au vu du nombre de sites visés par notre scraper (une dizaine) et du poids moyen d'un replay (quelques dizaines de ko), l'ensemble des composants de l'application pourront largement être hébergés sur une seule machine, tant du point de vue accès disque que bande

passante (réseau).

3. Aspect juridique

Après avoir vu les différents techniques et concepts de récupération de données sur internet, nous pouvons nous poser la question de la réglementation. En effet, même si le principe de politesse existe et implique la non dégradation des services par le scraper, par exemple, ce dernier est-il réellement autorisé à récupérer les données et surtout que peut-il en faire ?

Il faut bien comprendre que l'objectif d'un scraper ou d'un crawler n'est pas de voler des données. Le but est de récupérer, le plus souvent de façon périodique, les données d'un site. Les données en question sont majoritairement accessibles par un humain qui naviguerait sur la page web. Afin d'automatiser le processus, les scripts sont apparus. La masse d'information récupérée n'est donc pas illégale. Elle est même mise à disposition par le site. Cependant, K. A. Adler [16] explique que les entreprises ne peuvent pas laisser leurs concurrents les étudier aussi facilement.

Pour contrer ces attaques, les propriétaires de site rédigent des conditions d'utilisation strictes. De ce fait, le scraping doit être plus prudent et moins intrusif. Mais avez-vous déjà souvent accepté des conditions d'utilisation en naviguant sur un site? On remarque que dans l'affaire opposant *EF Cultural Travel* à *Explorica* [16], où *Explorica* ajustait ses prix 5% moins cher que *EF* en récupérant leurs tarifs, les *"terms of use"* n'ont pas été efficaces. C'est principalement parce que d'anciens employés ont utilisé une faille dans le code du site qu'ils ont été inculpés. La cour a également jugé que les accès par le scraper étaient supérieurs à un accès *"raisonnable"* d'un site classique. *Explorica* n'a jamais reçu d'avertissements sur ces accès de la part d'*EF*. Enfin il est complexe de calculer les pertes subies par EF. Dans un vol classique, l'objet est physiquement retiré à son propriétaire mais dans le cas des scripts, les données sont simplement copiées. Il est parfois difficile de différencier un script d'un utilisateur lambda. Pour conclure, les conditions d'utilisation d'un site doivent être claires et visibles pour être efficaces.

Il y a un manque certain dans la réglementation pour définir ce qui est autorisé et ce qui est interdit. Internet est un outil encore trop jeune et son développement est tellement rapide qu'il est impossible que la juridiction suive le rythme [16]. Dans une seconde affaire opposant *American Airlines* à *FareChase*, les premiers ont tout de même réussi à utiliser une loi sur la violation de propriété [17]. *FareChase* avait développé un logiciel qu'il vendait pour récupérer les informations sur les vols de *AA* et pouvoir proposer des offres concurrentielles. Après plusieurs avertissement de la part de *AA*, *FareChase* avait fait le choix de modifier leur logiciel pour masquer les accès. La cour a jugé que *FareChase* était un frein au bon développement du site de *AA* car il compliquait la maintenance du site et le logiciel touchait au système informatique de *AA* ce qui correspondait à une violation de la propriété. Les utilisateurs finaux étaient aussi impactés car le logiciel de *FareChase* entravait le service rendu par *AA*. *Facebook* avait utilisé la même loi lorsqu'un scraper récupérait des centaines

de comptes. Le scraper se défendait en indiquant que les données n'appartenaient pas à *Facebook* mais à ses utilisateurs et donc que la firme ne pouvait pas les accuser. La réseau social a donc contre-attaqué en prouvant que le scraper passait d'abord par le site de *Facebook* pour ensuite accéder aux comptes. On remarque bien dans ces deux procès que dénoncer un accès illégal est très complexe. Il est nécessaire de clarifier la situation.

Certaines entreprises qui sont sans défense d'un point de vue légal se tournent vers une protection technique. Impreva propose un outil qui détecte et bloque les accès automatisés. D'après leur livre blanc [5], SecureSphere permet de diminuer considérablement les attaques mais on peut se poser des questions sur l'efficacité de leur logiciel. Les utilisateurs ne sont pas patients et il ne faudrait pas que la protection entrave le bon fonctionnement du site. Tout comme les propositions de stratégie de défense faites par Impreva qui peuvent rendre un site web peu ergonomique. L'utilisation de CAPTCHA, par exemple, est souvent peu appréciée par les internautes.

Enfin, la réglementation est un peu plus présente dans le cadre de l'open data [15] car tout le monde cherche à collaborer. Il est cependant indispensable de vérifier la provenance des données et le cadre juridique dans lequel se place son propriétaire. Au sein des pays membres de l'UE, les données du secteur publique sont réutilisables pour un coût très faible ou nul depuis une directive de décembre 2011 puis un une proposition acceptée par tous de la Commission européenne. Un obstacle est à respecter sur les données nominatives qui sont encadrées par la loi Informatique et Liberté et qui de ce fait ne sont pas utilisables. Pour terminer, la finalité d'une action détermine le caractère publique d'une information : les données produites par un opérateur privé dans le cadre d'une mission de service publique, sont des données publiques, à l'exception des missions industrielles, commerciales et de sécurité nationale.

Conclusion

Le Web grossit à une vitesse folle et bien entendu, toutes les technologies qui l'entourent évoluent à la même vitesse. Les donnée sont aujourd'hui le nerf de la guerre. Les entreprises ont aujourd'hui compris qu'internet peut-être un moyen formidable d'expansion. Le crawling, le scraping, mais surtout l'object search et l'open data n'en sont qu'au début d'une croissance certaine. Il suffit d'observer le nombre de projet, d'articles et de conférences sur le sujet pour comprendre qu'un phénomène émerge. Le Web n'est, aujourd'hui, certainement pas exploiter à son maximum. Cependant, il faut savoir rester prudent car la quantité d'information est telle qu'il serait très facile de s'y perdre. Les futurs outils développés devront faire face à un nombre incalculable de données. On peut d'ores et déjà penser que structurer la mise à disposition de l'information comme le conseil le mouvement de l'open data par le biais d'API fournies sera un moyen efficace de gérer cette masse de données. Enfin, un travail est certainement à faire du côté de la réglementation pour clarifier la situation et permettre de se défendre face aux attaques illégales.

Bibliographie

[1] MARC NAJORK. Web Crawler Architecture [en ligne]. Mountain View, CA, USA : Microsoft Research, [2009]. Disponible sur http://research.microsoft.com/pubs/102936/eds-webcrawlerarchitecture.pdf (consulté le 04.11.2013)

[2] DATA PUBLICA. Crawling et Scraping : exploiter les données du Web pour développer votre business [en ligne]. 2013. Disponible sur : http://www.data-publica.com/content/2013/09/le-livre-blanc-de-data-publica-consacre-au-crawling-et-au-scraping (consulté le 10.10.2013)

[3] RICARDO BAEZA-YATES, CARLOS CASTILLO, MAURICIO MARIN, et al. Crawling a Country: Better Strategies than Breadth-First for Web Page Ordering [en ligne]. 2005. Disponible sur http://wwwconference.org/2005a/cdrom/docs/p864.pdf

[4] MICHAEL SCHRENK. Webbots, Spiders, and Screen Scrapers. No Starch Press, 2007. Disponible sur http://edu.ercess.co.in/ebooks/php/Webbots,%20Spiders,%20and%20Screen%20Scrapers%20-%20A%20Guide%20to%20developing%20internet%20agents%20with%20PHP.pdf
ISBN 978-593327-120-6, p407-416

[5] IMPREVA. Detecting and Blocking Site Scraping Attacks [en ligne]. 2011. Disponible sur http://www.imperva.com/docs/WP_Detecting_and_Blocking_Site_Scraping_Attacks.pdf

[6] SOUMEN CHAKRABARTI. Mining the web [en ligne]. Bombay, India : Morgan Kaufmann Publishers, 2003. Disponible sur http://read.pudn.com/downloads75/ebook/275001/Morgan%20Kaufmann%20-%20Mining%20the%20Web%20-%20Discovering%20Knowledge%20from%20Hypertext%20Data.pdf

[7] CARLOS CASTILLO. Effective Web Crawling [en ligne]. Santiago, RM, Chile : University of Chile, 2004. Disponible sur http://www.chato.cl/papers/crawling_thesis/effective_web_crawling.pdf

[8] SOUMEN CHAKRABARTI, MARTIN VAN DEN BERG, BYRON DOM. Focused crawling: a new approach to topic-specific Web resource discovery [en ligne]. Bombay, India : Indian Institute of Technology, 1999. Disponible sur http://www.cse.iitb.ac.in/~soumen/doc/www1999f/pdf/www1999f.pdf

[9] ANDREW PETERSON. BeautifulSoup : Web Scraping with Python [en ligne]. 2013. Disponible sur http://www.nyu.edu/projects/politicsdatalab/workshops/BeautifulSoup.pdf

[10] CHARLES SEVERANCE. Scraping Web Pages [en ligne]. University of Michigan : School of Information. Disponible sur https://ctools.umich.edu/access/content/group/8a0551bf-5d66-4a39-004b-3a90ff183423/Lectures/SI182%20-%20Scraping%20Web%20Pages.pdf

[11] CHRIS HANRETTY. Scraping the web for arts and humanities [en ligne]. Norwich, Royaume-Uni : University of East Anglia, 2013. Disponible sur http://www.essex.ac.uk/ldev/documents/going_digital/scraping_book.pdf

[12] ZAIQING NIE, YUNXIAO MA, SHUMING SHI, et al. Web Object Retrieval [en ligne]. Beijing, China : Microsoft Research, [year??]. Disponible sur http://research.microsoft.com/en-us/um/people/znie/fp626-nie.pdf

[13] KIM CUONG PHAM, NICHOLAS RIZZOLO, KEVIN SMALL, et al. Object Search: Supporting Structured Queries in Web Search Engines [en ligne]. University of Illinois : Department of Computer Science. Disponible sur http://cogcomp.cs.illinois.edu/papers/PRSCR10.pdf

[14] AURORE NICOL, JULIE CARUSO, ÉRIC ARCHAMBAULT. Open Data Access Policies and Strategies in the European Research Area and Beyond [en ligne]. Science-metrix, 2013. Disponible sur http://www.science-metrix.com/pdf/SM_EC_OA_Data.pdf

[15] PHILIPPE RICHARD. Open Data, une ouverture pour améliorer la société [en ligne]. 2013. Disponible sur http://pro.clubic.com/technologie-et-politique/article-560928-4-open-data-ouverture-ameliorer-societe.html

[16] KENNETH A. ADLER. Controversy Surrounds 'Screen Scrapers': Software Helps Users Access Web Sites But Activity by Competitors Comes Under Scrutiny [en ligne]. NEW YORK LAW JOURNAL, 26/03/2008. Disponible sur http://library.findlaw.com/2003/Jul/29/132944.html (consulté le 07.11.2013)

[17] AMERICAN LAW INSTITUTE, The Restatement (Second) of Torts, §217 Trespass to chattels.