Chapitre I : Introduction aux Graphes D'Interaction

1. Introduction

Depuis 1999, date à laquelle sont apparus les systèmes pair-à-pair (réseaux sociaux, la messagerie instantanée ou le Chat, systèmes de partage de fichiers, …etc.). Les systèmes P2P n'ont cessé de croître au point d'être aujourd'hui estimés à plusieurs dizaines de millions d'utilisateurs. Les réseaux pair-à-pair ne sont pas les seuls réseaux de cette importance, on rencontre dans différents domaines des réseaux d'interactions de très grande taille.

On appelle réseau d'interactions tout ensemble d'entités interagissant de façon individuelle. Les grands réseaux d'interactions recouvrent ainsi des réseaux aussi divers que le réseau des routeurs d'Internet, le réseau des contacts sociaux entre individus, ou le réseau des réactions chimiques entre protéines dans le métabolisme d'un être vivant. On parle également de réseaux réels. Des études récentes ont montré qu'en modélisant ces réseaux par des graphes, on observait des propriétés communes malgré leurs origines diverses (Internet, Web, Sociologie, Chimie, …etc.). La connaissance des propriétés des réseaux d'interactions est nécessaire afin de prévoir leur évolution et de déterminer leurs capacités à résister à différents phénomènes ou tout simplement de comprendre leur nature.

La modélisation de réseaux d'interactions consiste à construire des graphes aléatoires avec les mêmes propriétés que les réseaux réels. L'intérêt est de pouvoir effectuer sur machines des simulations de propagation, de pannes, d'attaque et d'autres événements qui peuvent survenir sur les réseaux réels. L'avantage de la modélisation est de pouvoir obtenir des graphes de grande taille ayant des propriétés réelles en temps raisonnable.

2. Définition d'un graphe d'interaction

Les graphes d'interaction sont une modélisation utilisée dans de nombreuses disciplines. A un instant de l'évolution du réseau, les acteurs sont les nœuds d'un graphe où une arête (arc) modélisent une interaction entre les deux nœuds qu'elle relie. Ce réseau est bien sûr dynamique, des nœuds peuvent être ajoutés ou supprimés durant son évolution. Les arêtes peuvent aussi être dynamiques selon la définition de l'interaction considérée [1].

Une interaction, dans un tel réseau, est définie selon ce que l'on cherche à modéliser, par exemple :

- Les réseaux issus de l'Internet : Un des réseaux les plus connu est le réseau Internet (ou encore les réseaux d'Internet). Plusieurs applications utilisent ce réseau pour transmettre de l'information, tel que « World-Wide-Web » ou encore « pair-à-pair ». Les instances de ces applications sont représentées par des nœuds, et les relations qu'elles entretiennent entre elles, peuvent être vues comme des arêtes.

- Les réseaux sociaux : Les graphes d'interactions sont utilisés couramment en sciences sociales pour modéliser des interactions entre individus. Les sommets représentent alors les entités ou individus, et les arêtes ou arcs une relation ou interaction entre eux.

- Les réseaux de cooccurrence lexicale : Le graphe de cooccurrence lexicale d'un document donné est un graphe dont les sommets sont des mots d'un document. Deux mots sont reliés s'ils apparaissent proches dans le document [2].

La figure suivante illustre un graphe connexe pondéré avec 379 sommets d'un réseau d'une collaboration scientifique autour de la thématique des réseaux sociaux [3].

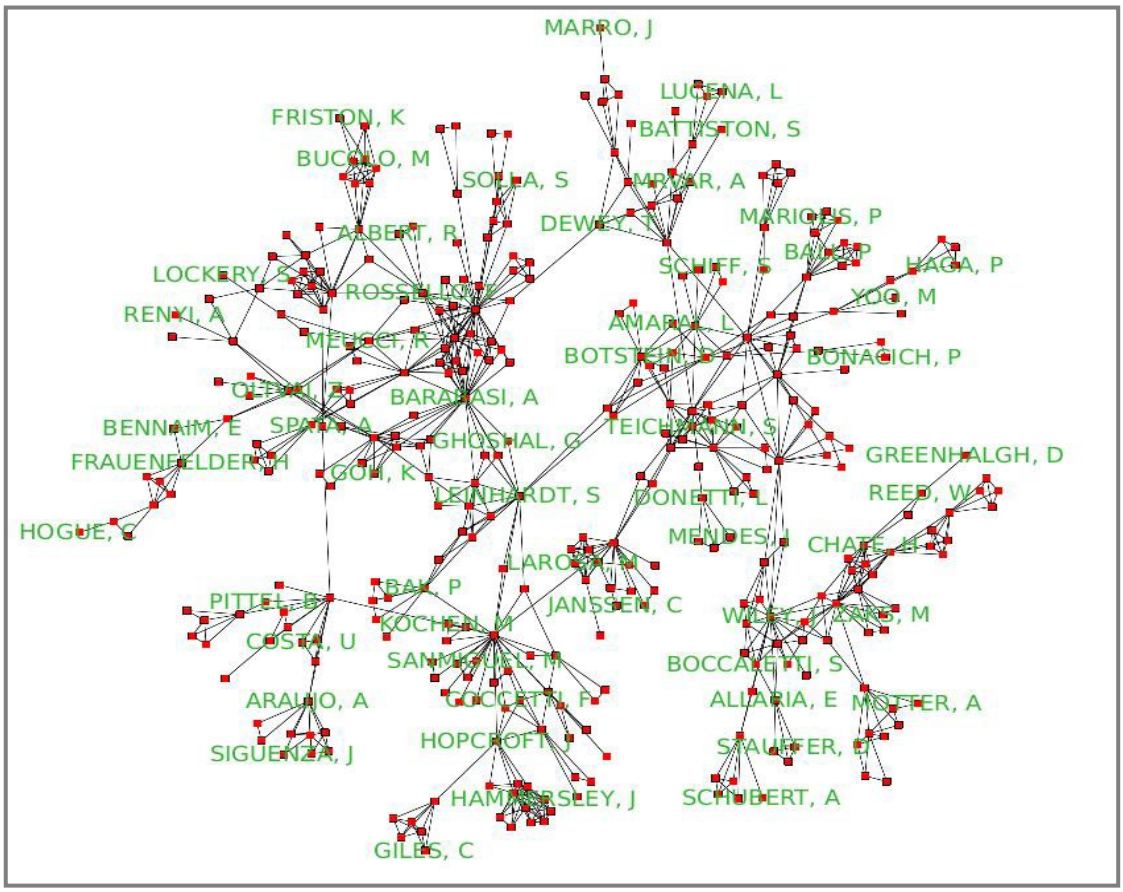


Figure I. 1: Un Graphe d'Interaction ou Réseau.

3. Propriétés communes des réseaux d'interactions

L'étude des réseaux d'interactions en tant que graphes a révélé que ces derniers, bien que issus de différents domaines, partagent quelques propriétés intéressantes. Dans cette section, nous présentons trois propriétés communes aux réseaux d'interactions :

- Distribution des degrés en loi de puissance,
- Distance moyenne faible (ou petit diamètre),

- Coefficient de regroupement (coefficient de clusterisation).

3.1 Distribution des degrés en loi de puissance

La distribution des degrés d'un réseau d'interactions suit une loi de puissance [4] de type

$$P(k) = C.k^{-\lambda} \quad (1)$$

Equation I. 1 : La loi de puissance.

Où : k est le degré, C est un paramètre qui dépend de la taille du réseau et λ est l'exposant de la loi de puissance. Dans la pratique la valeur de λ est comprise entre 2 et 3 et ne dépend pas de la taille du graphe.

Une distribution des degrés en loi de puissance signifie qu'il existe beaucoup de sommets de faible degré et très peu de sommets de fort degré. L'exposant représente la vitesse de décroissance de la courbe des degrés. Plus il est grand, plus la probabilité d'obtenir des sommets de fort degré est petite. Le degré moyen dans un graphe en loi de puissance n'est pas significatif car l'écart-type est très important. Les graphes en loi de puissances sont appelés graphes sans échelle (scale-free graphs).

3.2 Petit diamètre et distance moyenne faible

Le diamètre d'un graphe est le plus long des chemins parmi l'ensemble des plus courts chemins pour tout couple de nœuds dans le graphe. La distance moyenne est la moyenne des longueurs des plus courts chemins entre tous les couples de nœuds d'un graphe. Beaucoup ont pensé que le diamètre des réseaux d'interactions pourrait être grand et que leur distance moyenne pourrait être forte. Mais des études dans [5, 6, 7] sur différents réseaux d'interactions ont montré qu'il n'en est rien.

Les réseaux d'interactions ont un petit diamètre et une distance moyenne faible de l'ordre de log(n) où n est la taille du graphe. Le calcul de la distance moyenne et du diamètre étant coûteux en temps, une estimation de la distance moyenne est faite en l'évaluant seulement pour un certain nombre de paires de sommets.

3.3 Coefficient de regroupement fort

Le coefficient de regroupement (d'agrégation, de clusterisation ou de clustering) est une mesure de la connectivité d'un graphe non orienté, introduite en 1998 par Watts et Strogatz [8].

Soit d(u) le degré du sommet u ; le coefficient de regroupement du sommet u noté C(u) est défini par la formule suivante.

$$\mathrm{C(u)} = \frac{2*\text{nombre de connexion entre les } \textbf{voisins} \text{ de u}}{\mathrm{d(u)*(d(u)-1)}} \quad (2)$$

Equation I. 2 : Coefficient de regroupement.

Le coefficient d'agrégation C d'un graphe est égal à la moyenne des coefficients d'agrégation de l'ensemble de ses nœuds.

Des études dans [5, 6, 7] sur différents réseaux d'interactions ont montré que le coefficient d'agrégation de ces réseaux est élevé. Dans un réseau social, par exemple, cela signifie que les amis d'un même individu ont une grande probabilité d'être amis entre eux.

4. Modélisation des réseaux d'interactions

L'étude des grands graphes d'interaction a permis de découvrir les principales propriétés de ces graphes. De nombreux scientifiques ont dès lors cherché à trouver un modèle qui se rapprocherait le plus des grands réseaux réels. Historiquement, ce genre de graphe ont été d'abord assimilés à des graphes aléatoires mais de nombreuses différences subsistent.

Dans les points qui suivent nous allons tenter de présenter les principaux modèles proposés ainsi que les points de similarités et de divergences avec les réseaux réels.

4.1 Les Graphes aléatoires uniformes

Rien, à priori, ne relie les différents graphes réalistes. Ce qui amène à penser tout simplement que ces graphes ne sont que le résultat d'interconnexions établies au hasard entre les nœuds.

La théorie des graphes aléatoires a été introduite par Paul Erdös et Alfred Rényi après la découverte d'Edrös qui démontre que des méthodes probabilistes peuvent être très utiles pour venir à bout de certains problèmes dans la théorie des graphes.

La modélisation des grands réseaux d'interaction par des graphes aléatoires était donc une première tentative. Elle était considérée, jusqu'à récemment, comme la seule méthode, par défaut, pour les réseaux réels.

4.1.1 Le modèle aléatoire d'Erdös et Rényi

Lors d'une série de séminaires entre 1950 et 1960, Paul Erdös et Alfréd Rényi ont proposé et étudié les premiers modèles de réseaux d'interactions appelés les graphes aléatoires [8, 9, 10].

Erdös et Rényi ont apporté plusieurs versions de ce modèle, le plus étudié est construit comme suit :

1- Ajouter n sommets.

2- Pour tout couple de sommets, une arête est ajoutée avec une probabilité p.

Bilan de ce modèle :

Les récentes observations expérimentales (qui étaient impossibles ou très partielles auparavant) ont mis en lumière d'importantes différences. Ainsi, la distribution des degrés dans les graphes aléatoire d'Erdös et Rényi suit une loi de Poisson (exponentielle) alors qu'il s'agit d'une loi de puissance pour la grande majorité des réseaux réels. Ce graphe a un faible coefficient de clustering puisque les voisins d'un nœud n'ont aucune raison d'être davantage reliés entre eux que deux nœuds pris au hasard. Néanmoins, les graphes d'Erdös et Rényi partagent la caractéristique du petit diamètre avec les réseaux réels.

4.2 Les graphes petit-monde

Les graphes petit monde (Small World) sont nés après l'apparition du concept des « *six degrés de séparation* ». Le concept des six degrés de séparation a été imaginé par le Hongrois Frigyes Karinthy en 1929. Cette idée théorique évoque la possibilité que toute personne sur le globe peut être reliée à n'importe quelle autre à travers une chaîne de relations individuelles comprenant au plus cinq autres maillons.

Cette théorie fut reprise en 1967 par le psycho-sociologue Stanley Milgram qui a réalisé une série d'expériences dont le résultat est connu sous le nom de «six degrés de séparation». La conclusion de ses études est, que deux personnes quelconques dans le monde ont une chaîne de connaissances de longueur six en moyenne. En d'autres termes, il y a cinq personnes intermédiaires qui séparent les deux personnes étrangères l'une de l'autre. Par contre, après plus de trente ans, le statut de cette idée comme description de réseaux sociaux hétérogènes reste une question ouverte. Des études sont encore menées actuellement sur le « petit monde ».

4.2.1 Le modèle de Watts et Strogatz

Watts et Strogatz [11] proposent une méthode pour générer des graphes petits mondes. Ces graphes permettent d'interpoler le modèle d'un graphe ordonné et fini à un graphe aléatoire. Voici les étapes de construction du graphe selon ce modèle :

1- Construction d'un anneau régulier à n sommets où chaque sommet est relié à ses 2*k plus proches voisins (k voisins de chaque côté).
2- Randomiser : dans cette étape, on introduit une procédure (Random Rewiring) aléatoire de réorganisation des liens entre chacun des nœuds. Cette procédure fonctionne comme suit :
   - Pour chaque sommet et pour chaque arête, avec une probabilité p, l'arête est redirigée vers un autre sommet choisi aléatoirement et uniformément et avec une probabilité (1-p) l'arête est gardée.

La Figure I.2, montre que pour aller du régulier (Uniforme) à l'aléatoire total, on passe certainement par le small-world.

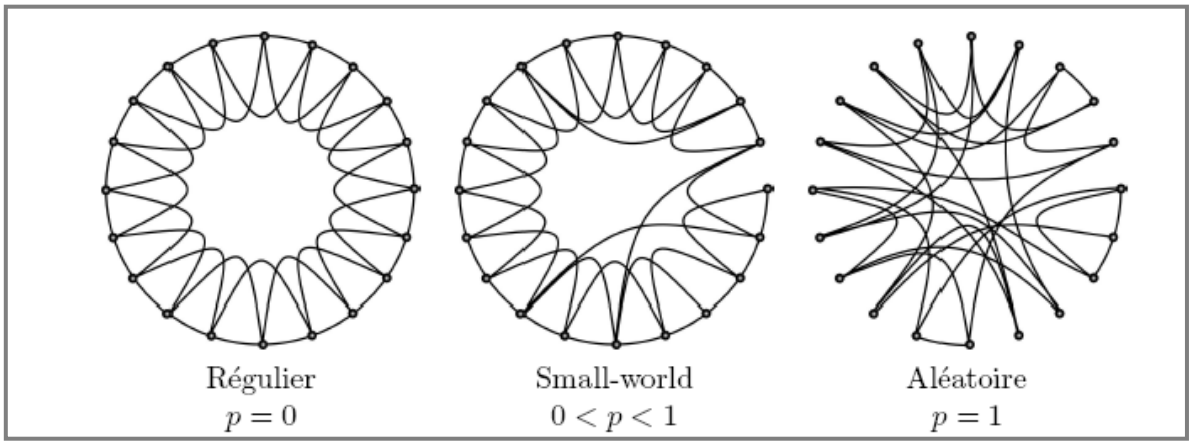


Figure I. 2 : Random rewiring procedure [12].

Bilan de ce modèle

Ce modèle théorique est très difficile à mettre en œuvre de manière décentralisée en raison de sa construction et de son coût de maintenance (initialisation, dynamisme difficilement applicable dû à sa méthode de construction, etc.). De plus, la distribution des degrés dans l'anneau de Watts et Strogatz suit une loi de poisson et non une loi de puissance. Cependant, en prenant une certaine valeur de p on obtient un graphe avec une distance moyenne faible et un fort coefficient de clusterisation.

4.2.2 Le modèle de Kleinberg

En 2000, Kleinberg [10] propose le premier modèle de petit monde présentant la propriété de navigabilité, c'est-à-dire le premier modèle de graphe dont le diamètre est poly logarithmique en nombre de nœuds et dont des chemins poly logarithmiques peuvent être découverts par un algorithme décentralisé entre tout couple de sommets.

Voici les étapes de construction du graphe selon ce modèle (voir Figure I.3):

1- Construire une grille à deux dimensions où chaque croisement dans la grille est un nœud, et chaque nœud est lié à ses quatre voisins immédiats : haut, bas, droit et gauche. Ces liens sont appelés « lien court ».

2- Ajouter à chaque nœud *u* un nombre k > 0 constant d'arc. La destination du j^{ème} arc de *u* est le nœud *v* avec une probabilité proportionnelle à $\frac{1}{|u-v|s}$, où $|u - v|$ est la distance entre *u* et *v* dans la grille, *s* : est une constante positive. Ces liens sont appelés « *liens longs* ».

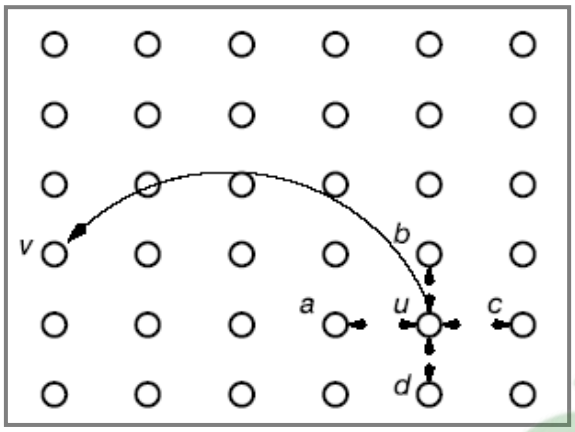


Figure I. 3 : Construction de réseau petit-monde: modèle de Kleinberg [13].

Bilan de ce modèle

L'idée de Kleinberg est de relier un sommet à tous ses proches voisins et à quelques sommets lointains. La première démarche permet d'augmenter la valeur du coefficient de regroupement et la deuxième de diminuer la distance moyenne. Le principal défaut de ce modèle est qu'il ne produit pas la distribution des degrés observée sur les réseaux réels, puisque chaque sommet a un degré constant.

4.3 Les réseaux scale-free

Les graphes aléatoires ont longtemps été utilisés pour modéliser les réseaux d'interactions. Des études récentes menées sur les réseaux d'interactions ont révélé que la distribution des degrés de ces derniers suit une loi de puissance. Or, la distribution des degrés dans les graphes aléatoires suit une loi de Poisson.

Depuis, plusieurs modèles aléatoires ont été proposés [14, 15] pour générer des graphes aléatoires avec une distribution des degrés en loi de puissance qui fait d'eux des graphes sans-échelle.

4.3.1 Le modèle de copie

Ce modèle génère des graphes orientés. Il est basé sur le comportement de l'utilisateur sur le Web. En effet, lorsqu'un utilisateur crée une page Web, il va relier cette dernière au reste du Web par des liens hypertextes. Pour cela, il va choisir des pages sur le Web (des prototypes). Un prototype est une page Web populaire et donc possédant de nombreux liens entrants. Le choix des prototypes n'est pas aléatoire mais plutôt en relation avec le contenu de la nouvelle page Web. Par la suite, l'utilisateur copie des liens hypertextes des prototypes dans la nouvelle page Web.

Kumar et al. [52] proposent deux modèles de copie : le modèle linéaire et le modèle exponentiel.

- Un graphe du modèle linéaire [14] est construit de la manière suivante :

1- A l'état initial, aucune information n'est donnée.

2- A l'étape t, un sommet *u* et *d*, arcs sont ajoutés. Les *d* arcs sont utilisés pour relier le sommet *u* au reste du graphe. Ensuite deux sommets *v* et *w* sont choisis aléatoirement tel que :

- avec une probabilité λ, le i^{éme} lien du sommet *v* est copié dans la page *u*,
- avec une probabilité 1 - λ, un lien est ajouté entre *u* et *w*.

✌ Un graphe du modèle exponentiel est construit de la manière suivante [14] :

1- A l'état initial, aucune information n'est donnée.

2- A l'étape *t*, *m* sommets et *l* arcs sont ajoutés tel que :

* *m* = Binomial ((nombre de sommets à l'étape *t-1*), *p*) avec $p > 0$, en moyenne $(l+p)t$.

* $l = (d + \gamma) * p *$(nombre de liens à l'étape $t - 1$) avec $\gamma > 1$.

3- Les l arcs sont utilisés pour relier les nouveaux sommets au reste du graphe de la manière suivante :

* avec une probabilité $(1 - \lambda)$, la destination est choisie aléatoirement parmi les *m* nouveaux sommets.

* avec une probabilité λ, la destination est choisie parmi les sommets déjà existants. La probabilité de choisir le sommet *v* dépend du degré sortant.

* $p > 0$, $\gamma > 1$, $\lambda \in [0, 1]$ et $d > 0$ sont des paramètres du modèle.

Bilan de ce modèle

Le modèle de copie permet de générer des graphes proches du graphe du Web. Dans le modèle linéaire, à chaque étape un seul sommet est ajouté, or, dans le Web plusieurs pages Web sont ajoutées en même temps (le graphe est très dynamique). Le modèle exponentiel permet d'ajouter à chaque étape un nombre de sommets qui dépend de la taille du graphe à un instant, et d'autres paramètres du modèle. Les inconvénients majeurs de ce modèle sont l'absence de la loi de puissance sur les degrés sortants des graphes générés et le fait d'avoir un faible coefficient de regroupement.

4.3.2 Le modèle à base d'attachement préférentiel de Barabàsi-Albert

En 1999, Albert et Barabasi [6] ont popularisé un modèle dynamique permettant d'obtenir une distribution des degrés suivant une loi de puissance. Ce modèle consiste à construire un

réseau nœud par nœud, en reliant chaque nouveau nœud préférentiellement aux sommets existants de plus hauts degrés. Il est connu sous le nom de l'attachement préférentiel.

Un graphe du modèle de Barabàsi-Albert (BA) est construit de la manière suivante :

1- n_0 sommets sont créés.

2- A l'étape t, un sommet est ajouté ainsi que m ($0<m\leq n_0$) arêtes pour le relier au reste du graphe. La probabilité qu'un sommet v soit choisi comme destination de l'arête est égale au degré de v sur le nombre d'arête à l'étape t-1.

Avec l'attachement préférentiel, un sommet choisi à l'étape t a une probabilité plus élevée d'être choisi à l'étape t+1.

Bilan de ce modèle

L'intérêt de ce modèle est sa construction dynamique, puisque dans de nombreux réseaux réels, des nœuds et des liens sont fréquemment ajoutés et enlevés au cours du temps (on peut penser au réseau des pages Web par exemple).

La construction d'un graphe suivant le principe de l'attachement préférentiel engendre la propriété de la distribution des degrés en loi de puissance. L'étude de ce genre de graphes a démontré que la distance moyenne entre les sommets est faible ce qui prouve bien que cette propriété est implicitement induite par les graphes présentant une distribution de degrés en loi de puissance.

Malgré les nombreux points en commun qui existent entre les graphes réels et les graphes obtenus avec ce modèle, ces derniers ne sont pourtant pas fidèles aux trois grandes propriétés connues des graphes réels puisque le coefficient de clustering reste relativement faible.

4.4 Les graphes bipartis

Un graphe est dit biparti s'il existe une partition de son ensemble de sommets en deux sous-ensembles U et V telle que chaque arête ait une extrémité dans U et l'autre dans V. Un graphe biparti permet notamment de représenter une relation binaire. Il peut aussi être représenté par un graphe uni-parti (voir Figure I.4) où deux sommets u et v de U sont reliés entre eux, s'il existe au moins un nœud de V, relié avec u et v. Le passage du graphe uni-parti au graphe biparti est impossible.

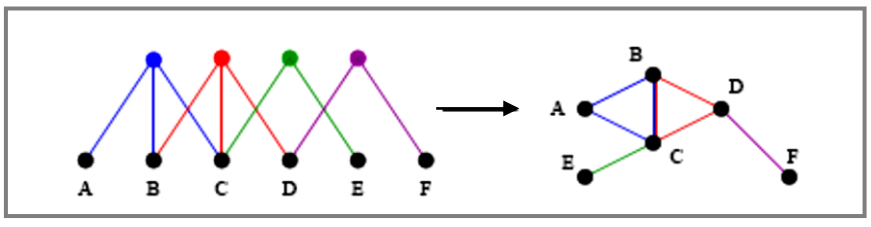


Figure I. 4: Un graphe uni-parti généré à partir d'un graphe biparti.

Clicours.COM

L'étude [16] portante sur le graphe biparti entre les auteurs et leurs publications, à partir de la base bibliographique DBLP4, montre que le graphe des coauteurs possède les propriétés

des graphes réels (distribution des degrés en loi de puissance, distance moyenne faible et fort coefficient de regroupement).

L'étude [17] s'est penchée sur le graphe des physiciens où deux physiciens sont liés s'ils ont partagé plusieurs ressources de calcul et de stockage ; le graphe des internautes ayant accédé au site web de Boeing où deux internautes sont liés s'ils ont visité un ensemble de pages en commun et le graphe des utilisateurs de Kazaa où deux utilisateurs sont liés s'ils partagent plusieurs fichiers en commun. Cette étude montre que ces graphes exhibent les mêmes propriétés que dans les graphes réels vus précédemment.

Les mêmes propriétés ont été observées dans l'étude [18] sur le graphe d'interactions des mots (où deux mots sont liés s'ils co-occurrent dans la même phrase). Une autre propriété observée dans les graphes bipartis est, l'abondance de biclique. Une biclique est un graphe biparti complet orienté où chaque sommet de U pointe (à un arc sortant) vers chaque sommet de V.

On peut tenter d'analyser la signification d'une biclique dans le Web comme suit : des pages Web d'un ensemble L pointent toutes vers des pages Web d'un ensemble W. Cela signifierait que les pages Web de L et W partagent des informations sur un sujet particulier. Dans ce cas, on dit que les pages Web ont un intérêt commun. Les deux ensembles L et W forment ce que Jon M. Kleinberg [10] définissent comme étant une communauté.

Kumar et al. [62] ont montré empiriquement que le Web contient un grand nombre de bicliques de différentes tailles.

4.4.1 Le modèle biparti de Guillaume et Latapy

Guillaume et Latapy [19] proposent deux modèles aléatoires pour les réseaux d'interactions qui peuvent être représentés naturellement par des graphes bipartis. Le premier modèle proposé est le modèle uniforme. Il consiste à générer des graphes bipartis aléatoires ayant des distributions de degrés données. La génération peut être faite de la manière suivante :

1- Créer les sommets de U et V et assigner à chacun un degré choisi suivant la distribution.
2- Dupliquer chaque sommet de U et V autant de fois que son degré.
3- Relier de manière aléatoire les sommets de U aux sommets de V.

Le deuxième modèle est le modèle à attachement préférentiel qui utilise l'attachement préférentiel pour relier les sommets de U aux sommets de V. Ce modèle a un paramètre λ appelé taux de recouvrement. Il varie d'un réseau à un autre et représente la proportion de sommets de V préexistants à laquelle un nouveau sommet de U est relié. L'étude [20] mesure en moyenne la valeur de λ sur des graphes bipartis réalistes. Elle aboutit à des valeurs comprises entre 0.75 et 0.95. Le processus de génération du graphe biparti peut être écrit comme suit :

- Ajouter un sommet *u* dans U et choisir son degré *d* en accord avec une distribution donnée.

- Pour chacun des *d* liens du sommet *u*, ajouter un lien vers un sommet de V selon l'attachement préférentiel avec une probabilité λ. Sinon (avec une probabilité 1-λ) créer un nouveau sommet dans V et le relier à *u*.

Bilan de ce modèle

Le modèle biparti de Guillaume et Latapy permet de générer des graphes avec un fort coefficient de regroupement qui ne dépend pas de la taille du graphe, une distance moyenne faible, une distribution des degrés en loi de puissance, et un grand nombre de biclique. Ce modèle respecte donc toutes les propriétés des graphes bipartis réels, sauf qu'il ne peut être dynamique.

5. Comparaison entre les différents modèles

Le Tableau I.1 montre qu'en règle générale, les modèles ne couvrent pas toutes les propriétés communes aux réseaux d'interactions. En effet, les premiers modèles aléatoires possèdent uniquement la propriété du petit diamètre, ils sont les moins appropriés pour modéliser les grands graphes réels. Les modèles des graphes small world et des graphes sans échelles, possèdent tous les deux, deux propriétés, les premiers ont un petit diamètre et un fort coefficient de clusturing, et les seconds ont une distribution des degrés suivant une loi de puissance et un petit diamètre. Le fait que les graphes sans échelles soient dynamiques les rend plus représentatifs des grands graphes réels.

Seul le modèle Biparti de Guillaume et Latapy regroupe les trois propriétés : loi de puissance, petit diamètre et fort coefficient de regroupement.

Modèle	Dynamique	Loi de puissance	Petit diamètre	Clustering
Erdös et Rényi	Non	Non	Oui	Non
Watts et Strogatz	Non	Non	Oui	Oui
Kleinberg	Non	Non	Oui	Oui
Copie	Oui	Pour les arcs entrant	Oui	Non
Barabàsi-Albert	Oui	Oui	Oui	Non
Biparti de Guillaume et Latapy	Non	Oui	Oui	Oui

Tableau I. 1 : Comparaison entre les différents modèles de graphes d'interaction.

6. Conclusion

L'étude des graphes d'interaction nécessite leur modélisation afin que leur analyse et détection de propriétés soient correctes. Plusieurs modèles existent pour modéliser ses grands réseaux, mais il n'existe pas encore de modèle conforme aux graphes réels. De nombreuses recherches sont menées pour trouver un modèle qui réunit à la fois les trois propriétés de ce genre de graphes à savoir :

- Une distribution des degrés en loi de puissance,
- Une distance moyenne faible,
- Un fort coefficient de clustering.

Guillaume et Latapy ont proposé un modèle de graphes d'interactions bipartis qui regroupe toutes les propriétés de ses graphes. Ils ont montré aussi que tous les graphes d'affiliations pouvaient être naturellement transformés en graphes bipartis.

Dans le chapitre suivant, nous allons présenter la définition ainsi que le processus du crawling afin de réaliser une collecte de pages web correctes et facile à étudier.

Chapitre II : État de l'Art des Crawlers

1. Introduction

Le Web, qui est caractérisé par sa grande taille et sa nature dynamique, nécessite une maintenance permanente spécialement pour les systèmes de recherche d'information. Si nous revenons aux années 90 le volume du Web était de quelques millions de pages passant à des milliards à l'heure actuelle, déjà Google répertorie environ 1000 milliards de pages sur le Web [w1]. Cette ample quantité de pages Web a sollicité les moteurs de recherches. Ces derniers sont influencés par la qualité (sa variété, sa pertinence, etc.) d'une page Web. Les Crawlers facilitent le processus de la recherche d'information en suivant les liens hypertextes dans les pages Web pour télécharger automatiquement et mettre à jour les nouvelles pages Web afin de les stocker dans une base de données où elles sont indexées pour répondre aux requêtes des utilisateurs.

Dans ce chapitre nous définissons ce qu'est un crawler et les cibles qu'il vise afin de connaitre les stratégies sur lesquelles il repose. Ces stratégies induisent des méthodes de parcours du graphe du Web pour en venir au modèle papillon. Etant donné que de nombreuses applications innovantes du Web crawling sont encore inventées, nous présentons brièvement quelques-unes qui ont déjà été développées.

2. Définition d'un Crawler

Le crawler est un logiciel, programme ou script informatique qui explore automatiquement le World Wide Web d'une manière méthodique et ordonnée. Il est généralement conçu pour collecter les ressources (pages Web, images, vidéos, documents Word ou PDF, etc.) afin de permettre à un moteur de recherche de les indexer.

Différentes appellations existent, certaines sont en langue anglaise Web crawler ou Web spider et d'autres sont en français explorateur du Web, robot d'indexation ou littéralement araignée du Web.

Le crawler est géré par un moteur de recherche pour construire un résumé du contenu d'un site Web (index de contenu) [21]. Le crawler crée un résumé textuel du contenu des adresses URL de chaque page du site (dictionnaire de mots). Donc il fait que l'indexation, du contenu textuel des sites Web par exemple des images (tels que des photos, graphiques, textes), des éléments vidéo et Flash nécessitent un HTML-Texte complémentaire pour être vu par un moteur de recherche.

En général, le crawler commence par une liste d'URL à visiter, appelées les graines. En visitant ces URL, le crawler identifie tous les liens hypertexte dans la page et les ajoute à la liste des URL à visiter, appelée la frontière crawl. Les URL de la frontière sont récursivement visitées selon un ensemble de règles.

Figure II. 1: Opération de Crawling.

Le grand volume de données implique que le crawler ne peut télécharger un nombre limité de pages Web dans un temps donné, il faut donner la priorité à ses téléchargements. Le taux élevé de changement des pages Web implique qu'elles aient déjà été mises à jour ou même supprimées, ceci crée un problème pour les crawlers car ils doivent trier une infinité de combinaisons de sélections HTTP GET basées sur l'URL afin d'avoir des changements relativement infimes de script pour récupérer un contenu unique.

Les moteurs de recherches utilisent les crawlers comme un moyen pour assurer la mise à jour des données et principalement pour créer une copie de toutes les pages visitées pour un traitement ultérieur, suite à une indexation les moteurs de recherches fournissent des recherches rapides.

Les crawlers peuvent également être utilisés pour automatiser les tâches de maintenance sur un site Web, telles que la vérification ou la validation des liens du code HTML. De plus, ils peuvent être utilisés pour recueillir certains types d'informations à partir des pages Web, telle que la récolte des adresses e-mail généralement pour envoyer du Spam.

Comme Edwards et al. [22] ont déclaré: «Étant donné que la bande passante pour la réalisation d'un crawl n'est ni infinie ni libre, il devient essentiel d'explorer le Web, non seulement de façon évolutive, mais efficace, si une mesure raisonnable de la qualité ou la fraîcheur doit être maintenu». Un crawler doit choisir avec soin à chaque étape les pages à la prochaine visite, ceci incite un certain comportement qui est le résultat d'un ensemble de politiques [23], nous citons :

- ✓ Une politique de sélection qui stipule les pages à télécharger,
- ✓ Une politique de revisite qui indique quand il faut vérifier les changements des pages,
- ✓ Une politique de politesse qui indique comment éviter la surcharge ses sites Web et
- ✓ Une politique de parallélisme qui indique comment coordonner des Web crawlers distribués.

Avant d'entamer les politiques il est préférable de faire un cours passage sur les applications et les stratégies du crawl.

3. Applications Crawling

Il y a différents scénarios dans lesquels les crawlers sont utilisés pour l'acquisition des données. Nous allons décrire dans la suite quelques exemples de stratégies utilisées dans le crawling.

3.1 Crawling en profondeur d'abord

Afin de construire un moteur de recherche important ou un grand dépôt comme l'archive d'Internet [18], les crawlers commencent avec un petit ensemble de pages, puis ils crawlent d'autres pages en suivant les liens dans un mode en "profondeur d'abord" (Voir chapitre IV, séction 4.2). En réalité, les pages web ne sont pas souvent parcourues dans un mode en profondeur d'abord stricte, mais en utilisant une variété de politiques, par exemple, pour élaguer les crawls à l'intérieur d'un site web ou pour crawler les pages les plus importantes en premier.

3.2 Recrawling des Pages pour des mises à jour

Une fois les pages sont initialement acquises elles peuvent être périodiquement recrawlées et contrôlées pour des mises à jour. Pour le faire, il faut soit commencer un autre crawl en profondeur d'abord ou tout simplement revisiter toutes les URLs collectées. Cependant une variété de méthodes heuristiques peuvent être utilisées pour recrawler les pages les plus importantes, les sites ou les domaines les plus fréquents. De bonnes stratégies de recrawling sont cruciales pour le maintien d'une mise à jour d'un index de recherche avec une bande passante limitée. Des travaux récents de Cho et Garcia-Molina [28] ont étudié les techniques d'optimisation de la «fraîcheur» de telles collections selon des observations données au sujet de l'histoire de mise à jour d'une page.

3.3 Crawling axé (ciblé)

Des moteurs de recherche plus spécialisés peuvent utiliser les politiques de crawling qui tentent de se concentrer uniquement sur certains types de pages. Par exemple, des pages sur un sujet particulier ou dans une langue particulière, des images, des fichiers mp3, ou de documents de recherche en sciences informatiques. En plus des heuristiques, des approches plus générales ont été proposées sur la base de l'analyse de structure des liens [10, 11] et les techniques d'apprentissage automatique [15, 24, 25]. L'objectif d'un crawler ciblé ou axé est de trouver de nombreuses pages d'intérêt sans utiliser beaucoup de bande passante. Ainsi, la plupart des travaux antérieurs n'utilisaient pas un crawler à haute performance, même si cela pourrait supporter de grandes collections spécialisées qui sont significativement à jour d'un moteur de recherche général.

3.4 La marche aléatoire et l'échantillonnage

Plusieurs techniques ont été étudiées qui utilisent des marches aléatoires sur le graphe du web pour échantillonner des pages ou estimer la taille et la qualité des moteurs de recherche [5, 16, 17].

3.5 Crawling le "Web Invisible"

Un grand nombre de données accessibles via le web réside en fait dans les bases de données et ne peut être récupéré en affectant des requêtes appropriées et/ou remplir des formulaires sur des pages Web. Récemment, un grand intérêt a porté sur l'accès automatique à ces données aussi appelé le « Web Invisible » ou « Web Profond », ou « Faits et chiffres fédérés ». Un travail dans [26] a étudié les techniques de crawling pour ces données. Un crawler tel qu'il est décrit ici pourrait être étendu et utilisé comme un frontal efficace pour un tel système. Notons, toutefois qu'il y a beaucoup d'autres défis liés à l'accès au Web invisible, et l'efficacité de l'extrémité frontale n'est probablement pas le problème le plus important.

4. Les politiques d'un crawl

4.1 La politique de sélection

Même les grands moteurs de recherche face à la taille actuelle du Web, ne couvrent qu'une partie de celle qui est accessible au public. Une étude en 2005 a montré que les moteurs de recherche qui passent à l'échelle n'indexent pas plus de 40% à 70% du Web indexable [27]. Alors qu'une étude précédente faite par le Dr Steve Lawrence et Lee Giles [28] a montré qu'aucun moteur de recherche n'indexe plus de 16% du Web en 1999. Comme un crawler ne télécharge toujours qu'une fraction des pages Web, il est hautement souhaitable que la fraction téléchargée contient les pages les plus pertinentes et pas seulement un échantillon aléatoire du Web.

Cela nécessite une métrique d'importance pour hiérarchiser selon la priorité les pages Web. L'importance d'une page est en fonction de sa qualité intrinsèque, sa popularité en termes de liens ou de visites et même de son URL (ce dernier est le cas des moteurs de recherche verticaux limités à un seul domaine de premier niveau, ou les moteurs de recherche restreints à un site Web fixe). Concevoir une politique de sélection a bien une difficulté supplémentaire: il faut travailler avec des informations partielles, comme l'ensemble des pages Web qui ne sont pas connues pendant l'analyse.

Cho et al. [29] ont fait la première étude sur les politiques pour l'exploration de planification. Leur ensemble de données est une exploration de 180.000 pages à partir du domaine stanford.edu, dans lequel une simulation de crawling a été faite avec des Stratégies différentes. Les paramètres de commande testés sont : en largeur d'abord, par comptage des backlinks et des calculs PageRank partielles. Une des conclusions était que si le crawler veut télécharger des pages avec un PageRank élevé au début du processus de crawling, alors que la stratégie « PageRank partielle » est la meilleure, suivie de « largeur d'abord » et « calculs backlinks ». Cependant, ces résultats sont à un seul domaine.

Najork et Wiener [30] ont exécuté un crawl réel de 328 millions de pages. En utilisant la largeur d'abord,ils ont constaté qu'un crawl en largeur d'abord capture les pages avec un PageRank élevé au début de l'exploration (mais ils n'ont pas comparé cette stratégie avec d'autres stratégies). L'explication donnée par les auteurs de ce résultat est que «les pages les plus importantes ont de nombreux liens vers de nombreuse hôtes. Ces liens seront détectés tôt, quel que soit l'origine du crawl hôte ou page ».

Boldi et al. [31, 32] ont utilisé la simulation d'un sous-ensemble du Web de 40 millions de pages à partir du domaine « *.it* » et 100 millions de pages de l'exploration WebBase. Les tests de cette simulation ont été lancés en largeur d'abord, en profondeur d'abord, en ordre aléatoire et avec une stratégie omnisciente. La comparaison a porté sur la façon dont le PageRank est calculé sur un crawl partiel qui se rapproche la valeur du vrai PageRank. Surprenant, certaines visites qui accumulent un PageRank très rapidement (notamment, en largeur d'abord et la visite omnisciente) fournissent des approximations progressives très pauvres.

Baeza-Yates et al. [33] ont utilisé la simulation sur deux sous-ensembles du Web de 3 millions de pages à partir du domaine .gr et .cl, en testant plusieurs stratégies de crawling. Ils ont montré que la stratégie de TOPIC et la stratégie qui utilise la longueur des files d'attente par site sont meilleures que le crawling en largeur d'abord, et qu'il est aussi très efficace d'utiliser un crawl précédent, quand il est disponible, afin de guider le crawl actuel.

Daneshpajouh et al. [34] ont conçu un algorithme basé sur la communauté pour découvrir les bonnes graines. Leur méthode explore les pages Web avec un PageRank élevé de différentes communautés avec moins d'itération dans la comparaison avec un crawl qui commence à partir des graines aléatoires. On peut extraire une bonne semence dans un graphe Web précédemment crawlé à l'aide de cette nouvelle méthode. L'utilisation de ces graines peut améliorer une nouvelle opération de crawling.

4.2 Crawling axé (Focused crawling)

L'importance d'une page pour un crawler peut être également exprimée en fonction de la similitude d'une page pour une requête donnée. Les Web crawlers qui tentent de télécharger des pages qui sont semblables les unes aux autres sont appelés crawlers axés ou crawlers topiques. Les concepts de crawl topique et ciblé ont été introduits par Menczer [35] [36] et par Chakrabarti et al. [37].

Le principal problème pour un crawler axé est que dans le contexte d'un Web crawler, il est préférable qu'on soit en mesure de prédire la similitude du texte d'une page donnée à la requête avant son téléchargement. Un indicateur possible donné par le texte d'ancre des liens, ce qui est l'approche adoptée par Pinkerton [38] dans le premier Web crawler des premiers jours de l'Internet. Diligenti et al. [39] proposent d'utiliser le contenu complet des pages déjà visitées pour déduire la similarité entre la requête en cours et les pages qui n'ont pas encore été visitées. La performance d'un crawler axé dépend essentiellement de la richesse des liens de la rubrique spécifique qui est recherché, et un crawler topique repose généralement sur un moteur de recherche Web général pour fournir des points de départ.

4.2.1 Restriction des liens suivis

Un Crawler peut seulement chercher des pages HTML et éviter tous les autres types MIME, pour *Multipurpose Internet Mail Extension*. MIME est un format de message Internet permettant de découper un message en plusieurs parties et d'y inclure des données non-ASCII, à savoir du son des images.

Afin de demander les ressources HTML, un crawler peut faire une requête HTTP HEAD pour vérifier si la ressource Web est de type MIME. Pour éviter de faire de nombreuses requêtes HEAD, un crawler peut examiner l'URL et demander uniquement, si l'URL se termine par certains caractères tels que .html, .htm, .asp, .aspx, .php, .jsp, .jspx ou un slash. Cette stratégie peut entraîner involontairement l'omission de nombreuses ressources Web HTML.

Certains crawlers peuvent aussi éviter de demander toutes les ressources qui ont un "?" (Elles sont dynamiquement produites) afin d'éviter les pièges d'araignée (Spider traps) qui peuvent causer le téléchargement d'un nombre infini d'URL d'un site Web. Cette stratégie n'est pas fiable si le site utilise un moteur de réécriture pour simplifier son URL.

4.2.2 Normalisation d'URL

Généralement les crawlers effectuent un certain type de normalisation d'URL afin d'éviter le crawl de la même ressource plus d'une fois. Le terme de normalisation d'URL, également appelée canonisation d'URL, désigne le processus de modification et de normalisation une URL de manière cohérente. Il existe plusieurs types de normalisation qui peut être effectué, y compris la conversion des URL en minuscules, suppression des "." et segments, et en ajoutant des slashs au composant du chemin non vide [40].

4.2.3 Crawling du chemin ascendant

Quelques crawlers ont l'intention de télécharger des ressources autant que possible à partir d'un site Web particulier. Le crawling du chemin ascendant a été parcoure et crawle tous les chemins de chaque lien URL en montant vers sa racine. Par exemple, étant donnée l'URL suivante : « http://www.objectif-photos.net/photos/assekrem/algerie.htm » ; il va essayer de crawler « /photos/assekrem/ », « /photos/ » et « / ». Cothey [41] a constaté qu'un crawler d'un chemin ascendant était très efficace pour trouver des ressources isolées ou les ressources pour lesquelles aucun lien entrant n'aurait été trouvé pour un crawling régulier.

Beaucoup de crawlers de chemins ascendants sont également connus en tant que logiciel de la récolte du Web, car ils sont habitués à «récolter » ou recueillir tout le contenu à partir d'une page spécifique ou de l'hôte.

4.3 Politique de Revisite

Le Web a une nature très dynamique, et le crawling d'une fraction du Web peut prendre des semaines ou des mois. Au moment où un Web crawler termine son exploration, de nombreux événements peuvent se produire, y compris des créations, des mises à jour et des suppressions.

Du point de vue d'un moteur de recherche, il y a un coût associé à la non-détection d'un événement, et donc avoir une copie à jour de la ressource. Les fonctions de coût les plus utilisées sont la fraîcheur et l'âge [24].

- Fraîcheur: Il s'agit d'une mesure binaire qui indique si la copie locale est exacte ou non. La fraîcheur d'une page *p* dans le référentiel à l'instant *t* est définie comme suit :

$$F_p(t) = \begin{cases} 1 & \text{si } p \text{ est une copie égale à l'instant } t \\ 0 & \text{Autres} \end{cases} \quad (3)$$

Equation II. 1 : Fraîcheur d'une page.

- Age : Ceci est une mesure qui indique si la copie locale est à jour. L'âge d'une page *p* dans le référentiel à l'instant *t* est défini comme suit :

$$A_p(t) = \begin{cases} 1 & \text{si } p \text{ est modifiée à l'instant } t \\ t - \text{temps de modification de la page } p & \text{sinon} \end{cases} \quad (4)$$

Equation II. 2 : Age d'une page.

Coffman et al. [42] ont travaillé avec une définition équivalente à la fraîcheur, mais avec une formulation différente. Ils proposent qu'un crawler doit minimiser la fraction de temps des pages expirées. Ils ont également noté que le problème pour crawler le Web peut être modélisé comme une file d'attente multiple d'un système de requêtes à un serveur unique sur lequel le crawler est le serveur Web et les sites Web sont les files d'attente. Dès l'arrivée des clients, des modifications de pages sont faites et le basculement de temps est l'intervalle entre les accès aux pages d'un site Web unique. Selon ce modèle, le délai d'attente moyen pour un client dans le système de requête est équivalent à la moyenne d'âge pour le moteur de recherche Web.

L'objectif d'un crawlers est de garder la plus élevé possible la moyenne de fraîcheur des pages dans sa collection, ou de garder l'âge moyen des pages aussi bas que possible. Ces objectifs ne sont pas équivalents : dans le premier cas, le crawler est juste préoccupé par le nombre de pages qui sont expirées, alors que dans le second cas, le crawler est préoccupé par la connaissance d'âge des copies locales des pages.

Deux simples politiques de revisite ont été étudiées par Cho et Garcia-Molina [43] :

Politique uniforme : il s'agit de revisiter toutes les pages de la collection avec la même fréquence, indépendamment de leurs taux de variation.

Politique proportionnelle : il s'agit de revisiter le plus souvent les pages qui changent fréquemment. La fréquence de visite est directement proportionnelle à la fréquence (estimée) de changement.

Cliccours.com

Dans les deux cas, l'ordre répété de crawling des pages peut se faire soit dans un ordre aléatoire ou un ordre fixe.

Cho et Garcia-Molina [43] ont prouvé qu'en termes de fraîcheur moyenne, la politique uniforme surclasse la politique proportionnelle à la fois dans un crawl d'un Web simulé et d'un Web réel. Les crawlers Web ont une limite au nombre de pages qu'ils peuvent crawler dans un laps de temps donné, cela est pour les raisons suivantes :

(1) Vu l'évolution rapide des pages Web, le crawler va allouer un trop grand espace pour les nouveaux crawlings au détriment des mises à jour des pages de la collection, et

(2) La fraîcheur de l'évolution rapide des pages dure une période plus courte que celle avec un changement moins fréquent. En d'autres termes, une politique proportionnelle alloue davantage des ressources pour crawler les pages fréquemment mises à jour, mais elle connaît moins de durée de fraîcheur globale.

Pour améliorer la fraîcheur, le crawler devrait pénaliser les éléments qui changent trop souvent. La politique optimale de revisite est ni uniforme ni proportionnelle. La méthode optimale pour maintenir une grande fraîcheur moyenne inclut l'ignorance des pages qui changent sans cesse et la méthode optimale pour maintenir un âge moyen bas est d'utiliser des fréquences d'accès de façon monotone croissante selon la vitesse de variation de chaque page. Dans les deux cas, la méthode optimale est plus proche de la politique uniforme de la politique proportionnelle : comme Coffman et al. [44] ont noté, « afin de minimiser le temps d'abandon prévu, les accès à une page particulière doivent être maintenus aussi régulièrement espacés que possible".

Les formules explicites pour la politique de revisite en général ne sont pas réalisables, mais elles sont obtenues numériquement, car elles dépendent de la distribution des changements de page. Cho et Garcia-Molina [45] montrent que la distribution exponentielle est un bon ajustement pour décrire les changements de pages, tandis que Ipeirotis et al. [46] montrent comment utiliser les outils statistiques pour découvrir les paramètres qui influent cette distribution. A noter que les politiques de revisite considèrent toutes les pages les plus homogènes en termes de qualité (toutes les pages du Web ont la même valeur) chose qui n'est pas réelle, donc d'autres informations au sujet de la qualité d'une page Web doivent être incluses pour obtenir une meilleure politique de crawling.

4.4 Politique de politesse

Les crawlers peuvent récupérer des données beaucoup plus rapidement et plus profondément qu'un être chercheur, afin qu'ils puissent avoir un impact dévastateur sur les performances d'un site. Inutile de dire que, si un crawler unique effectue plusieurs requêtes par seconde et/ou le téléchargement de gros fichiers, toutefois un serveur aurait du mal à suivre les demandes de multiples crawlings.

Comme l'a noté Koster [47], l'utilisation de Web crawler est utile pour un certain nombre de tâches, mais elle est livrée avec un prix pour la communauté en général. Les coûts d'utilisation de Web crawlers incluent :

- Les ressources réseau : comme les crawlers nécessitent beaucoup de bande passante et ils fonctionnent avec un degré élevé de parallélisme au cours d'une longue période de temps ;
- La surcharge du serveur : surtout si la fréquence des accès à un serveur donné est trop élevé ;
- Les crawlers mal composés : qui peuvent planter des serveurs ou des routeurs ou qui téléchargent des pages Web non supportées et qu'ils ne peuvent pas gérer,
- Crawlers personnels, perturbent le réseau et les serveurs Web puisqu'ils sont déployés par de nombreux utilisateurs.

Une solution partielle à ces problèmes est le protocole d'exclusion des crawlers, aussi connu comme le protocole robots.txt qui est une norme pour les administrateurs pour indiquer quelles parties de leurs serveurs Web ne doit pas être accessible par les crawlers [48]. Cette norme ne comporte pas de suggestion pour l'intervalle de visites sur le même serveur, même si cet intervalle est le moyen le plus efficace pour éviter une surcharge du serveur. Récemment, les moteurs de recherche commerciaux comme Ask, Jeeves, MSN et Yahoo sont en mesure d'utiliser un supplément nommé "Crawl-delay: Délai-crawl ", un paramètre dans le fichier robots.txt pour indiquer le nombre de secondes pour le délai entre les demandes.

Le premier intervalle proposé entre les connexions est de 60 secondes [49]. Toutefois, si les pages ont été téléchargées à ce rythme, à partir d'un site web avec plus de 100.000 pages sur une connexion parfaite avec une latence égale à zéro et une bande passante infinie, il faudrait plus de 2 mois pour télécharger un seul site Web entier ; également, une seule fraction des ressources en provenance de ce serveur Web doit être utilisée. Cela ne semble pas acceptable.

Cho [50] Utilise 10 secondes autant qu'un intervalle des accès, alors que le crawler WIRE [51] utilise 15 secondes comme valeur par défaut de crawling. Le crawler MercatorWeb [52] suit une politique d'adaptation de politesse, la suivante :

- Si un téléchargement d'un document à partir d'un serveur donné prend t secondes, le crawler attend (10 t) secondes avant de télécharger la page suivante. Dill et al. [53] utilisaient une seconde.

Pour ceux qui utilisent les Web crawlers pour des recherches objectives, une analyse plus détaillée est nécessaire et des considérations déterminantes doivent être prises en compte au moment de décider ou crawler et à quelle vitesse il faut crawler [54].

Des témoignages de journaux d'accès montrent que les intervalles d'accès de crawling connues varient entre 20 secondes et 3-4 minutes. Il est intéressant de crawler que même en étant très poli, et en prenant toutes les mesures de protection pour éviter de surcharger les serveurs Web, des plaintes d'administrateurs de serveurs Web sont reçus. Brin et Page [55] notent que : "... Exécuter un crawler qui se connecte à plus d'un demi-million de serveurs (...) génère une quantité considérable d'appels e-mail et de téléphone. Puisque un grand nombre de

personnes qui viennent en ligne, on trouve toujours ceux qui ne savent pas ce que c'est un crawler, car c'est le premier qu'ils ont vu".

4.5 Politique de parallélisme (Exploration distribuée du Web)

Un crawler parallèle est un crawler qui exécute plusieurs processus en parallèle. L'objectif est de maximiser la vitesse de téléchargement, tout en minimisant les frais généraux de parallélisme et d'éviter les téléchargements répétés de la même page. Pour éviter de télécharger la même page plus d'une fois, le système d'exploration nécessite une politique visant à affecter les nouvelles URLs découvertes au cours du processus de crawling, comme la même URL qui peut être trouvée par deux processus différents de crawling.

5. Exigences pour un crawler

Nous allons maintenant discuter les exigences et les approches pour un bon crawler.

Flexibilité : il est préférable d'utiliser dans le système une variété de scénarios, avec moins de modifications possibles.

Un coût faible et une haute performance : le système doit évoluer pour atteindre au moins plusieurs centaines de pages par seconde, des centaines de millions de pages par cycle d'exécution et devrait fonctionner sur du matériel peu coûteux. Une utilisation efficace pour l'accès au disque est cruciale afin de maintenir une vitesse élevée auprès des structures de données principales, telles que les "URLs visitées" la structure et la frontière crawl deviennent trop grandes pour la mémoire centrale. Cela se produira seulement après le téléchargement de plusieurs millions de pages.

5.1 Robustesse

Il y a plusieurs aspects ici. Tout d'abord, puisque le système va interagir avec des millions de serveurs, il doit tolérer un code HTML erroné, comportements et configurations anormaux du serveur et de nombreux autres problèmes. Le but ici est d'ignorer avec prudence des pages et des serveurs qui ont un comportement étrange, puisque dans de nombreuses applications, il est possible de télécharger un sous-ensemble des pages. Deuxièmement, étant donné qu'une analyse peut prendre des semaines, voire des mois, le système doit être capable de tolérer les accidents et les interruptions de réseau sans perdre (trop) les données. Ainsi, l'état du système doit être gardé sur disque. Les propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité) ne sont pas vraiment demandées.

5.2 L'étiquette et le control de vitesse

Il est extrêmement important de suivre les conventions standards d'exclusion pour les crawlers (Robots.txt et les robots avec les méta-tags), pour fournir une URL de contact pour le crawler, et de superviser le crawl. En outre, on doit être en mesure de contrôler la vitesse d'accès de plusieurs manières différentes. On doit éviter de mettre trop de charge sur un

serveur unique, en contactant chaque site une seule fois toutes les 30 secondes. Il est également souhaitable de ralentir la vitesse au niveau du domaine, afin de ne pas surcharger les petits domaines, ainsi de contrôler la vitesse de téléchargement total du crawling si c'est une connexion partagées par plusieurs utilisateurs.

5.3 Gestion et reconfiguration

Une interface appropriée est nécessaire pour surveiller le crawl, y compris la vitesse du crawl, les statistiques sur les hôtes et les pages, et les tailles des ensembles de données principales. L'administrateur doit être en mesure de régler la vitesse, ajouter et supprimer des composants, éteindre le système, forcer un verrouillage, ou ajouter des hôtes et des domaines d'une «liste noire» des liens que le crawler doit éviter. Après un accident ou un arrêt, le logiciel du système peut être modifié pour résoudre les problèmes, et l'on peut vouloir continuer l'exploration en utilisant une configuration de machine différente.

6. Les algorithmes de recherche dans le Web

Les moteurs de recherche sur Internet doivent faire face à de très fortes contraintes. Les moteurs de recherche doivent répondre à des millions de requêtes par jour alors que la quantité de documents qu'ils doivent analyser est gigantesque (à cette date Google [w1], le moteur de recherche le plus populaire, indexe près de 1000 milliards de pages Web). La détection d'une mise à jour de ces documents est elle aussi source de grandes difficultés. Il n'y a pas de centralisation de ces données et un moteur de recherche doit par conséquent scruter en permanence le réseau en vue de détecter ces changements. Cette problématique est bien illustrée par une citation de Sergey Brin, un des inventeurs de Google et de l'algorithme PageRank [55] : «Le Web est une vaste Collection de documents hétérogènes complètement incontrôlés».

Pour toutes ces raisons, des recherches ont été menées afin d'alléger la charge des machines composant les systèmes de recherche. Le principal facteur de ralentissement étant l'analyse des textes afin d'établir un tri de pertinence de résultats, d'autres voies ont été explorées. Les meilleurs résultats ont été obtenus par des méthodes utilisant les propriétés déduites de l'analyse de la topologie d'Internet.

Dans les passages suivants nous allons examiner dans un premier temps les caractéristiques d'Internet et les documents qui composent ce réseau. Ensuite nous explorons les algorithmes de détection de pertinence employés dans les moteurs de recherche.

6.1 La topologie d'Internet

De nombreuses études ont été réalisées à la fin des années 1990 dans le but de déterminer et de caractériser les connexions établies par les liens hypertextes des documents d'Internet. Ce réseau diffère fondamentalement des autres réseaux connus de par sa nature ouverte et son

utilisation grandissante. Le développement d'Internet et plus particulièrement du World Wide Web est complètement incontrôlé. Chaque individu peut créer son propre site Web avec le nombre de documents et de liens hypertextes qu'il souhaite. Cette situation favorise une augmentation continue du nombre de pages et de liens disponibles et mène à la création d'un vaste réseau très complexe.

Il existe principalement deux manières de modéliser la topologie d'Internet sous forme d'un graphe. La première consiste à considérer les nœuds de routage du réseau comme les nœuds du graphe et les liaisons entre les routeurs comme des arcs non orientés. Cette modélisation a un intérêt dans l'optimisation des algorithmes de routage d'un réseau [56] mais peu du point de vue de la recherche d'information.

La seconde s'intéresse non pas aux connexions physiques du réseau mais aux relations qui existent entre les documents d'Internet. Cette modélisation représente Internet par un graphe orienté dont les nœuds et les liens sont respectivement représentés par des documents, un contenu textuel, une structure, ... et les URLs des liens hypertextes qui relient un document à un autre.

La topologie de ce graphe détermine la connectivité d'Internet et par conséquent nous renseigne sur la meilleure manière de rechercher de l'information dans ce réseau. Cependant, sa taille gigantesque, estimée à plusieurs milliards de nœuds et sa perpétuelle évolution tant au niveau des arcs que des nœuds, rend impossible l'analyse de l'ensemble de ces éléments.

Par conséquent, la plupart des travaux réalisés dans ce domaine ont tenté de déduire les lois régissant ce réseau à partir d'un sous-graphe d'Internet. Ces lois sont ensuite vérifiées et validées par des simulations de réseaux à plus grande échelle afin de déduire les propriétés et les caractéristiques du graphe complet d'Internet.

Une des premières constatations réalisée par l'étude de ce graphe est que plusieurs lois de puissance régissent Internet. Une loi de puissance est une expression de la forme $y \propto xa$ [57, w2], où a est une constante, x et y sont des mesures effectuées sur le sujet et $\propto$ signifie proportionnel à.

Faloutsos et Siganos ont déterminé trois lois de puissance caractérisant Internet au niveau des connexions existantes entre les serveurs de données [57]. Leur raisonnement a montré qu'il existait deux niveaux régissant Internet selon des paramètres différents. On distingue ainsi un haut niveau où plusieurs nœuds de communication s'entre-connectent et un plus bas niveau, à l'intérieur de chaque nœud, où plusieurs serveurs de données sont fortement reliés entre eux. La figure II.2 illustre cette situation. Les expérimentations ont été réalisées sur plusieurs sous-graphes d'Internet répartis en fonction des sous-domaines constatés et du graphe global.

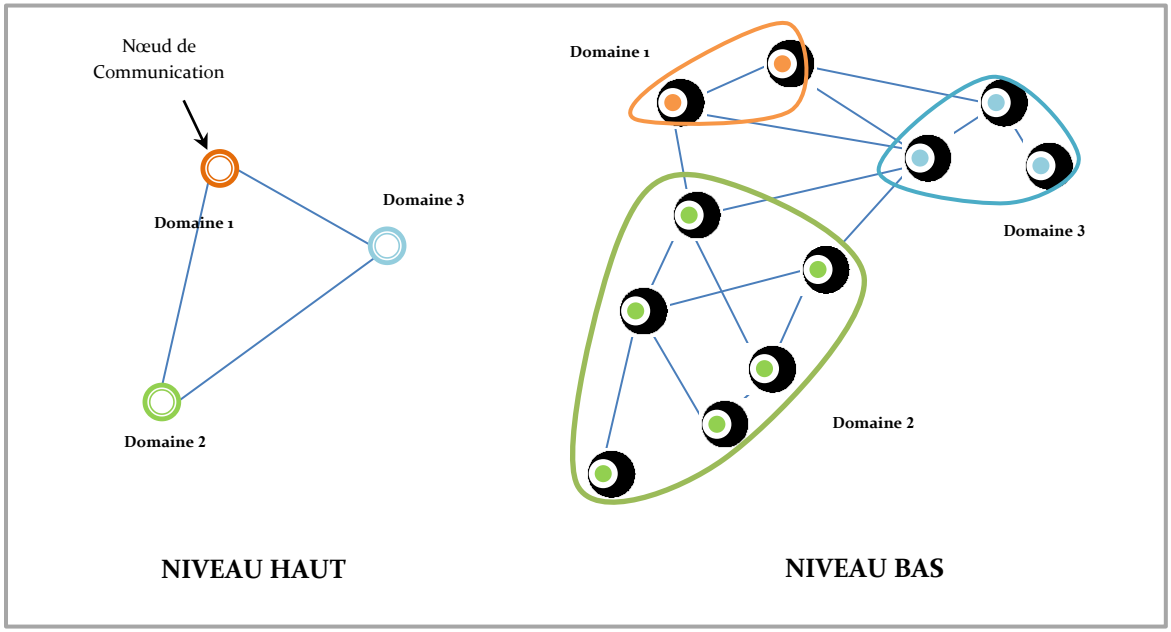


Figure II. 2: Structure d'Internet à deux niveaux.

Dans le graphe orienté représentant Internet, chaque nœud possède des liens sortants et des liens entrants. La loi de puissance cherche à caractériser l'organisation de ces liens. Pour cela, les nœuds sont triés dans l'ordre décroissant de leur nombre de liens sortants afin de leur attribuer un rang. Voir l'équation (5) et (1) du chapitre précédant.

$$d_v \propto r_v^R \tag{5}$$

Equation II. 3 : Le nombre de liens sortants d'un nœud.

Cette loi peut exprimer que « Le nombre de liens sortants d_v d'un nœud v est proportionnel au rang de ce nœud r_v à la puissance d'une constante R », selon les expérimentations R représente deux valeurs R ≈ -0.80 si on ne considère que les inter-domaines (liaisons internes à un domaine en particulier) et R ≈ -0.48 si on considère le graphe complet des liaisons. Ceci traduit le coût d'ajout d'un arc E en connaissant le nombre total de nœuds N, en fonction de la constante R comme le décrit l'équation (6).

$$E = \frac{1}{2(R+1)}\left(1 - \frac{1}{N^{R+1}}\right)N \tag{6}$$

Equation II. 4 : Coût d'ajout d'un arc.

La loi de puissance s'intéresse aussi aux valeurs propres des graphes d'Internet. Cette caractéristique est importante puisqu'elle traduit plusieurs propriétés topologiques d'un graphe comme son diamètre, le nombre d'arcs, l'arbre de couverture, le nombre de composantes connectées et le nombre de chemins d'une certaine taille entre les nœuds du graphe [58].

Le réseau se compose en sous réseaux connectés entre eux. Cela représente bien l'historique de construction d'Internet qui a, à son origine, servi à relier plusieurs réseaux entre eux. L'esprit de ce développement existe encore et est similaire au niveau de la construction des réseaux de documents que l'on peut séparer en plusieurs sites (regroupant chacun de nombreuses pages Web fortement connectées entre elles) reliés par des liens.

Internet est un réseau de routeurs connectés par des liens. Chaque routeur dépend d'une autorité d'administration ou de systèmes autonomes. Le développement des routeurs est fortement corrélé à la densité de population à travers le monde. Ce n'est pas une coïncidence puisqu'une forte densité de population implique une forte demande de services Web, résultant en une forte densité de domaines Internet. Yook [59] a montré dans son étude que le développement du réseau lui-même suit des propriétés fractales (Il a des détails similaires à des échelles arbitrairement petites ou grandes). Autrement dit, plus un nœud du graphe représentatif d'Internet possède d'arcs entrants, plus de nouveaux nœuds auront tendance à se lier à lui. Cette notion est très importante car elle est à la base de la plupart des algorithmes de détermination de pertinence d'une page dans les moteurs de recherche. En effet, on peut considérer que si un nœud du graphe est pointé par beaucoup de liens, celui-ci possède une caractéristique intéressante. Il agit comme une sorte d'autorité au regard d'autres nœuds plus faiblement connectés. Ainsi, lors de la création d'un nouveau point dans le graphe, la connexion se fera préférentiellement vers une autorité du domaine et donc vers un nœud fortement relié dans le graphe afin d'obtenir une certaine pertinence.

L'interrogation de la topologie de connexion des pages Web entre elles amène des informations très importantes sur la manière de rechercher de l'information dans les documents présents sur le Web. La vaste distribution des pages Web à travers le monde entier oriente vers une première problématique : comment les documents sont-ils reliés entre eux et comment les atteindre tous ? Le diamètre de notre graphe est ainsi une variable prépondérante du problème. Il permet de connaître le nombre de liens nécessaires à suivre pour atteindre n'importe quel point du graphe depuis un nœud donné. Albert, Jeong et Barabâsi se sont attachés à ce problème et ont levé le voile sur plusieurs de ces interrogations dans [60, 61].

Pour déterminer la connectivité des pages du Web, un crawler parcourt le graphe du Web en stockant dans une base de données toutes les URLs (les liens) rencontrées dans chaque document. Dans leur étude, les auteurs ont ainsi construit un sous-graphe d'Internet comportant près de 326 000 documents reliés par environ 1 470 000 liens. Les mesures effectuées sur cet ensemble de données ont établi que les probabilités qu'un document ait respectivement k liens entrants et k liens sortants (distribution de degrés), suivent une loi de puissance décrite dans le chapitre I, voir l'équation (1).

En effet, on retrouve au niveau des documents la même organisation en plusieurs niveaux avec la topologie observée au plan des connexions des routeurs et des serveurs d'Internet. Ainsi, plus une page sera pointée par d'autres pages, plus celle-ci aura tendance à être pointée par de nouvelles pages. Intuitivement, une page Web nouvellement créée inclura plus

facilement des liens vers des pages Web connues - et donc populaires - avec une très forte connectivité. La loi de puissance indique également que la probabilité de trouver un document à l'aide d'un grand nombre de liens est significative tant que la connectivité des pages du réseau est relativement importante. Et d'une manière symétrique, la probabilité de trouver des pages populaires - ou pointées par un grand nombre de documents - est non négligeable.

La loi de puissance régissant la connectivité et donc la topologie de ce graphe a permis d'établir une mesure du plus court chemin reliant deux documents au sein d'Internet. Cette mesure caractérise ce que l'on désigne par diamètre du Web et définit le nombre moyen de liens qu'il est nécessaire de suivre pour naviguer d'un document à un autre. Beaucoup ont pensé que le diamètre du graphe du Web pourrait être grand et leurs distances moyennes pourraient être fortes, mais des études ont montré que ce n'est pas vrai. Le graphe d'Internet a un petit diamètre et une distance moyenne faible de l'ordre de log(N) ; où N est le nombre des nœuds du graphe. L'équation suivante établie expérimentalement, indique le diamètre du web:

$$l = 0.35 + 2.06\ log(N) \qquad (7)$$

Equation II. 5 : Diamètre du Web.

Cela montre que le Web n'est en réalité qu'un petit monde extrêmement interconnecté, comme un système social ou biologique (le modèle Small World, le modèle d'attachement préférentiel, le modèle de Watts et Strogatz, modèle aléatoire d'Erdüs et Renyi,...). En considérant qu'Internet comporte environ 5 milliards de pages Web indexées, (l) = 20.33. On Peut ainsi naviguer d'un point de notre graphe de 5 milliards de nœuds à n'importe quel autre point du graphe en suivant 21 liens seulement en moyenne. Cette faible valeur de (l) nous permet de dire qu'un agent de recherche intelligent peut trouver n'importe quelle information rapidement en naviguant simplement sur le Web.

On retrouve dans la littérature [62, 63, 64, 65], la plus grande étude topologique d'Internet réalisée à ce jour comporte environ 200 millions de pages et 1,5 milliards de liens. Elle a été effectuée à partir de plusieurs crawls du moteur de recherche Altavista entre mai et octobre 1999 [64]. La première conclusion importante de ces travaux a conforté l'hypothèse que la distribution des liens entrants suit une loi de puissance. La valeur de λ a été vérifiée sur plusieurs échelles du graphe du Web et est restée constante pour $\lambda = 2.1$. D'un autre côté, le nombre moyen de liens sortants observé dans les crawls [62, 63] est de 7,2.

L'analyse des 200 millions de nœuds du graphe a également montré qu'environ 90% des pages forment un noyau fortement connecté si on ne tient pas compte de la nature orientée des liens. Il existe ainsi environ 17 millions de pages qui sont complètement dissociées de la structure fortement connectée que forme Internet. Une analyse plus précise a dénoté quatre familles de pages aux propriétés distinctes illustrées sur la figure II.3.

Quatre groupes se détachent : un noyau fortement interconnecté (SCC), un groupe de pages pointant vers le noyau (IN) et un groupe de pages pointées par le noyau (OUT).

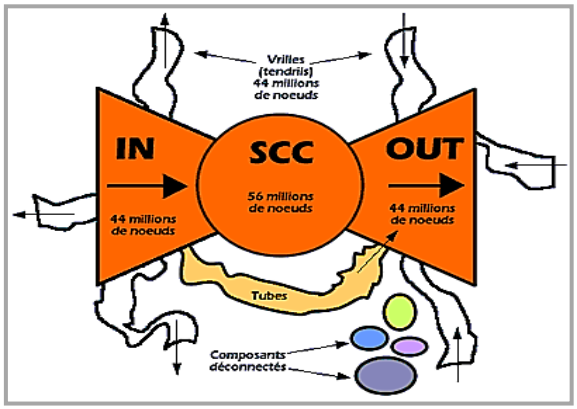


Figure II. 3 : Représentation du graphe de connexion des pages du Web.

Une première zone centrale dénommée SCC se détache du reste du graphe et détient 27,2% de l'ensemble des pages. Il s'agit des pages fortement connectées entre elles. Chacune peut atteindre une autre page en suivant quelques liens hypertextes. On peut aisément considérer ce noyau comme le cœur du Web, il représente 1/4 des pages, où la longueur moyenne d'un plus court chemin est comprise entre 16 et 20. Les deux familles suivantes (IN et OUT) ont une nature opposée mais incluent 21,5% de la totalité des pages. IN, appelée aussi Origine représente des pages qui peuvent atteindre le noyau mais qui ne peuvent pas être atteintes par lui (les arcs étant orientés).

Au contraire, OUT ou extrémité inclut les documents accessibles par les pages de SCC mais qui n'ont pas de lien retour, comme par exemple les sites de grandes compagnies qui ne contiennent pratiquement que des liens internes à leur site Web. Les vrilles (branches) contiennent des pages qui ne peuvent atteindre le noyau et réciproquement ne peuvent pas être atteintes par lui, l'ensemble des branches est de 21,5% des pages, et enfin les composantes déconnectées représentent 8,3%.

Une étude de [66] a montré que le diamètre du noyau tourne au tour de 28 nœuds, la distance moyenne entre la page de zone origine et une page de la zone extrémité en passant par le noyau est proche de 900, ainsi que la probabilité d'avoir un chemin entre deux pages quelconques est de 0,24.

Ces analyses statistiques nous permettent de mieux comprendre l'organisation des documents dans le Web afin de rendre des algorithmes de recherche les plus efficaces possible. Actuellement, deux algorithmes d'analyse des liens contenus dans les différents documents se sont imposés grâce à leur grande efficacité. Il s'agit de HITS et de Page-Rank.

6.2 L'Algorithme HITS (Hypertext Induced Topics Search)

Jon Kleinberg fut un des premiers à s'intéresser aux propriétés de connectivité du graphe représentatif d'Internet et de son apport dans la détection de la pertinence d'une page à une requête [67]. Quelques constatations simples sont à l'origine de ses travaux dans ce domaine. L'importance d'une page peut être extraite de la structure des liens du Web. Dans cette

approche, deux types de page sont identifiés en fonction de la nature de leurs connexions avec les autres documents. On retrouve d'une part les pages qui semblent être très importantes et jouent le rôle d'autorité sur un sujet donné et d'autre part les documents possédant un grand nombre de liens vers des pages faisant autorité sur un sujet.

On distingue ainsi les pages autorités ayant un grand nombre de liens entrants et les pages hubs ayant un grand nombre de liens sortants et regroupant les autorités d'un même sujet.

Le but de l'algorithme HITS est de déterminer les hubs et les autorités qui renforcent leurs relations mutuellement sur un sujet donné. Ainsi Kleinberg dénombre les bons hubs comme des pages pointant vers beaucoup de bonnes autorités et les bonnes autorités comme des pages pointées par beaucoup de bons hubs. Cette dénotation de bons hubs et de bonnes autorités fait apparaître une troisième catégorie de pages ayant un grand nombre de liens entrants provenant de documents n'ayant aucune particularité. Ces pages, que nous nommons pages indépendantes, sont considérées comme universellement populaires et n'apportent que peu ou pas d'intérêt [68]. Par exemple la page de Google est extrêmement référencée mais n'apporte que peu d'intérêt sur la plupart des sujets rencontrés ou une publicité aura de nombreux liens vers elle. Cette situation est illustrée dans la figure II.4 [77].

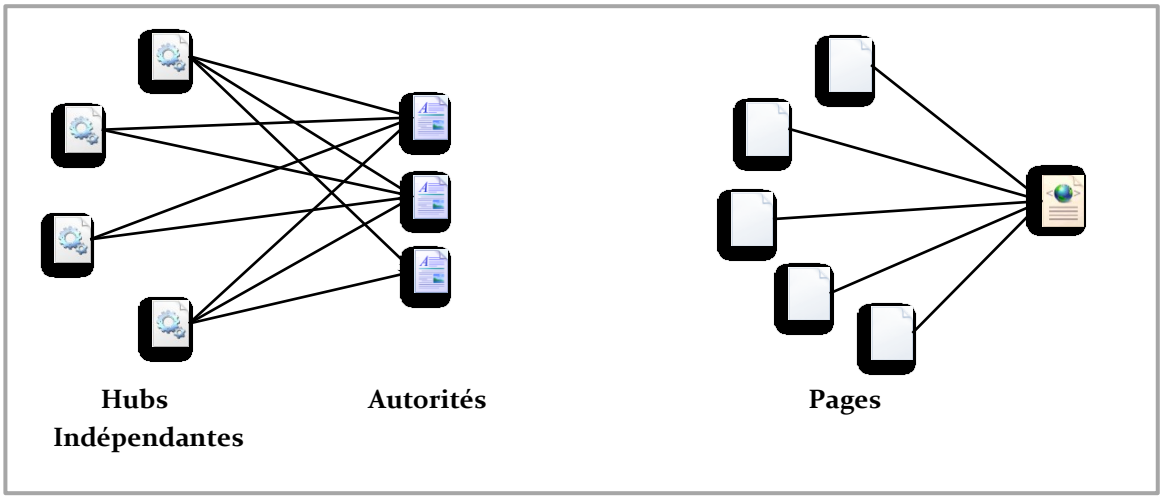


Figure II. 4 : Représentation des pages en bon Hubs, Bonnes Autorités et pages indépendantes.

Chakrabarti et al. estiment que la page, vers laquelle un lien est construit, évoque un sujet similaire et paraît intéressante. Pour déterminer les hubs et les autorités d'un sujet σ donné, l'algorithme HITS se base sur un sous-graphe d'Internet Sσ qui doit répondre aux conditions suivantes :

- Sσ est relativement petit,
- Sσ est riche en pages pertinentes,
- Sσ contient la totalité (ou la plupart) des plus importantes autorités.

Clicours.COM

En gardant Sσ petit, occupe moins de temps de calcul nécessaire à la réalisation de la

tâche. Les deux derniers points permettent d'avoir de bonnes chances de déterminer les bonnes autorités correspondantes à la requête σ.

Pour construire un tel graphe, l'algorithme utilise une requête de mots-clés (σ) afin de prendre en compte un petit nombre de pages (environ 200) depuis un moteur de recherche traditionnel. Cependant, cet ensemble de pages, noté Rσ, ne contient pas nécessairement l'ensemble des autorités du sujet σ. Par exemple, il y a de fortes chances pour que les réponses à la requête du moteur de recherche ne correspondent pas aux grands sites de moteurs de recherche : ces pages ne contiennent en effet que très rarement les mots-clés recherchés. On peut toutefois espérer que ce sous-graphe contienne des liens vers des autorités ou des hubs importants. Rσ est alors étendu en ajoutant les nœuds pointés par le sous-graphe et les nœuds pointant le sous-graphe afin d'obtenir un graphe augmenté Sσ. L'auteur affirme alors que les trois conditions requises par le sous-graphe Sσ sont respectées.

Les bonnes autorités et les bons hubs peuvent être extraits de ce graphe en donnant une définition numérique à la notion de hub et d'autorité. L'algorithme HITS associe un vecteur de potentiels d'autorités non négatifs x et un vecteur de potentiels de hubs non négatifs y pour toutes les pages appartenant à Sσ. Ainsi, une page p possédant un fort potentiel d'autorité x_p, respectivement un fort potentiel de hub y_p, sera vue comme étant une bonne autorité, respectivement un bon hub.

Initialement, aucune hypothèse n'est faite sur les potentiels hub et autorité de chaque document. Ainsi les vecteurs originaux x et y sont uniformes. Les poids sont ensuite mis à jour de la façon suivante : si une page est pointée par plusieurs bons hubs, son potentiel d'autorité est incrémenté. Ainsi, pour une page p donnée, la valeur de son poids x_p est mise à jour par la somme des potentiels y_q de toutes les pages q pointant sur p. D'une manière symétrique, les potentiels de hub sont mis à jour en fonction des potentiels d'autorités des pages vers lesquelles un document en cours pointe. Ces opérateurs sont décrits dans la figure II.5.

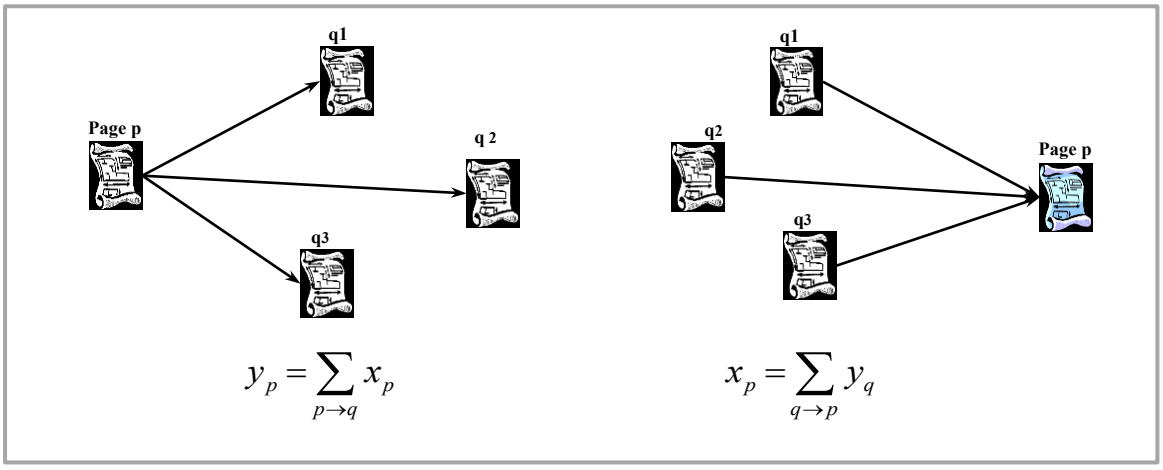


Figure II. 5: Opérateurs de mise à jour des accumulateurs Hub (yp) et Autorité (xp).

L'algorithme HITS est composé de deux listes ordonnées décroissantes de pages en fonction des potentiels respectifs de hub et d'autorité. L'originalité vient du fait que seule la topologie des liens entre les documents est prise en compte afin de déterminer les pages les plus pertinentes sur un sujet donné. Les expérimentations réalisées par Kleinberg montrent que sur une requête pour un moteur de recherche basé uniquement sur un index inversé, les résultats obtenus sont très satisfaisants : la requête « Search Engine » (posée en 1999) produit les sites Yahoo !, Excite, Magellan, Lycos et Altavista comme autorités bien que leur texte ne contienne pas la requête initiale.

La première grande implémentation de HITS fut réalisée au travers le projet CLEVER (Clientside Eigenvector Enhanced Retrieval : extraction améliorée de vecteur propre côté client) conçu au centre de recherche IBM Almadén et décrits dans [69]. CLEVER a apporté des améliorations sensibles à l'algorithme HITS, car ce dernier se base uniquement sur le nombre de liens, certaines pages se voient attribuer des notes artificiellement hautes, parce qu'elles reçoivent des liens de la part de hubs qui sont en fait des copies de la même page (un résultat bien connu le spamming). Dans le projet CLEVER, des filtres sont ajoutés pour ne pas tenir compte de liens dupliqués et des corrections sont apportées en fonction du contexte d'apparition du lien dans la page (les termes recherchés sont-ils proches ou éloignés du lien).

La mise à jour des potentiels maintenant prend en compte une nouvelle pondération sur les sommes en fonction du texte des pages concernées. Ces poids sont basés sur le nombre d'apparitions des termes n(t) de la requête pour chaque lien exploré. La zone d'exploration autour de chaque lien a été déterminée de manière statistique à 50 caractères de part et d'autre du lien. Ainsi, les arcs du graphe Sσ sont dorénavant pondérés par :

$$w(p,q) = 1 + n(t) \tag{8}$$

Equation II. 6 : Poids des Liens.

L'algorithme HITS repose sur un sous-graphe d'Internet afin de réduire les temps d'exécution des algorithmes. Ce sous-graphe est obtenu à partir d'une requête donnée. Le calcul de pertinence est réalisé pour chaque requête parvenue au moteur de recherche afin qu'elle se fasse sur le graphe entier. Il existe cependant des méthodes indépendantes de la requête proposée par l'utilisateur qui peuvent déterminer une pertinence absolue des documents du réseau.

Actuellement, le moteur de recherche Ask.com tourne autour de l'algorithme HITS.

6.3 L'Algorithme du Tri Global, PageRank

L'idée principale de cette méthode est de simuler le comportement d'un internaute naviguant de manière aléatoire sur Internet. La probabilité qu'il visite une page donnée est d'autant plus grande que cette page est pointée par beaucoup d'autres pages au travers de leurs liens hypertextes. En considérant qu'une page confère une certaine autorité à une autre page

en établissant un lien vers elle, la probabilité de passage de cet internaute aléatoire sur une page indique le degré de pertinence de ce document. Le problème avec cette méthode c'est que l'internaute ne distingue pas si le lien établit est vers une page de bonne ou de mauvaise qualité, puisque l'importance est exprimée par le nombre de liens pointant vers cette page.

Page et Brin ont tenu compte de ce problème, ils calculent le PageRank de chaque page Web en donnant un poids à chaque lien. Ce dernier est proportionnel à la qualité de la page contenant ce lien. Pour déterminer la qualité de la page donnée, le Pagerank est calculé d'une manière récursive avec une valeur initiale quelconque. Des heuristiques de modifications du poids de chaque lien ont notamment été étudiées dans [69] et [70].

Il existe deux manières d'accéder à une page Web sur Internet. On peut d'une part l'atteindre directement en connaissant son adresse et d'autre part suivre un lien hypertexte d'un autre document. Le calcul du PageRank (et donc de la pertinence) d'une page intègre ces deux éléments au travers d'une probabilité α ; qui représente en quelque sorte la probabilité qu'un internaute aléatoire s'ennuie sur une page et décide de choisir une autre page au hasard. L'équation II.7 permet de calculer le PageRank pour tous les nœuds du graphe représentant Internet.

$$R(p)_t = (1-\alpha) + \alpha \sum_{(q,p)\in G^2} \frac{R(q)}{d^+(q)} \quad (9)$$

Equation II. 7 : PageRank.

Dans cette formule, $R(p)_t$ représente la valeur du PageRank à l'itération t pour la page p . $d^+(q)$ est défini comme le nombre de liens sortants de la page q. Le paramètre α prend ses valeurs dans l'intervalle [0-1] et est généralement placé à α = 0.85 d'après des études statistiques menées par Lary Page dans [71], ce dernier paramètre permet de faire converger l'algorithme de manière plus ou moins rapide. En effet, plus α est élevé, plus l'effet de l'ajout d'un lien entrant vers une page est accru et plus celui-ci se propagera dans toutes les pages d'un même site.

Le PageRank forme ainsi une distribution de probabilités des pages Web. Le calcul peut s'effectuer de manière itérative et converge vers une valeur asymptotique (se dit d'une ligne ou d'une surface dont une autre ligne ou une autre surface s'approche indéfiniment, sans pouvoir jamais la toucher.) de manière assez rapide. En effet, le calcul effectué dans [71] sur un graphe de 26 millions de nœuds en considérant α = 0.85, converge en seulement 52 itérations.

Dans la formule, pour chaque itération toutes les pages distribuent leur PageRank en fonction de leurs liens sortants, la distribution (une distribution uniforme) s'effectue de manière plus ou moins importante en fonction du nombre de liens $d^+(q)$ de chaque page q du graph G. Cela peut aisément s'interpréter en considérant qu'une page possède un certain potentiel de pertinence et que le concepteur de la page donne une certaine crédibilité (son PageRank) aux pages pour lesquelles il inscrit un lien.

La formule initiale suppose que la probabilité de surfer sur n'importe quelle page est uniforme. Mais une des plus importantes variations de cet algorithme consiste à attribuer une distribution de probabilité non uniforme afin de mieux caractériser la réalité d'Internet [72]. L'équation II.7 peut être reformulée de manière matricielle. Pour cela, on considère le vecteur *r* qui correspond à la loi de distribution de l'ensemble des nœuds du graphe et une matrice *M* stochastique telle que l'entrée M[p, q] = 0 s'il n'existe pas de lien sortant de la page *q* vers la page *p* et $M[p,q] = \frac{1}{d^+(p)}$ sinon.

Donc l'équation

$$R(p) = \frac{\alpha}{N} + (1-\alpha) \sum_{(q,p)\in v^2} \frac{r(q)}{d^+(q)}$$

prend ainsi la forme suivante :

$$R = M^t \times r + \left[\frac{\alpha}{N}\right] \quad (10)$$

Equation II. 8 : Formule de PageRank de manière matricielle.

Où Mt désigne la transposée de la matrice *M*. La sommation de l'équation II.8 peut être représentée par la multiplication matricielle $M^t \times r$.

Deux problèmes avec l'approche du PageRank sont soulignés par Kleinberg dans [73].

Premièrement, plusieurs pages peuvent être considérées comme des autorités particulièrement importantes sur des sujets donnés mais ne contenant pas les termes de la requête correspondante. Cela pose un problème de par la méthode de sélection des pages utilisée, la sélection des réponses pertinentes se fait simplement à l'aide d'un index inversé et le tri de ces pages est effectué grâce au score obtenu par le PageRank. Les solutions obtenues contiennent alors forcément les termes de la requête.

Un autre problème est souligné concernant les pages qui ont un nombre de liens entrants très important comme par exemple www.yahoo.com. Ces pages auront un PageRank très élevé. Cependant, si la requête proposée contient un mot compris dans ces pages mais qui n'a pas de rapport avec la requête, la page va être considérée comme autorité sur le sujet donné. Kleinberg conclut d'ailleurs en disant que l'utilisation unique de la structure des liens entrants ne permet pas d'obtenir un équilibre correct entre la notion de popularité et la notion de pertinence.

Afin de corriger ces éventuels problèmes, plusieurs auteurs ont tenté d'effectuer une hybridation entre les algorithmes PageRank et HITS. Citons par exemple HubRank [74] qui modifie le vecteur de personnalisation de distribution de probabilité *r* de l'équation II.8 en prenant en compte à la fois la distribution des liens sortants et des liens entrants du graphe de connexion des pages Web. Le surfer aléatoire va alors préférer de naviguer sur une page ayant un fort degré de liens sortants avec la probabilité (1 - α).

Un autre exemple d'amélioration intéressante est donné dans [75, 77]. Les auteurs ont cherché à combiner les avantages de l'algorithme HITS avec l'algorithme PageRank. Il utilise la notion de parcours aléatoire du graphe d'Internet présent dans l'algorithme HITS permettant de détecter les pages aux caractéristiques de hubs et d'autorités avec l'initialisation du surfeur aléatoire présent dans la première partie de l'équation du PageRank II.8. Les résultats obtenus par les auteurs montrent alors que ce nouvel algorithme est plus robuste aux perturbations rencontrées dans les différents graphes de test.

6.4 L'Algorithme SALSA (Stochastic Approch for Link Structure Analysis)

Un algorithme alternatif, SALSA, a été proposé par Lempel et Moran en 2000 [76]. Son but est similaire à celui de l'algorithme HITS décrit dans la section 6.2. Il cherche à déterminer les meilleurs pages correspondantes à un sujet donné en les caractérisant en hubs et autorités.

La méthode SALSA pour résoudre ce problème est toutefois différente de celle utilisée par Kleinberg dans l'algorithme HITS. Elle consiste à parcourir au hasard les nœuds du graphe d'Internet en suivant les liens les reliant avec une distribution de probabilité uniforme. Les pages les plus visitées correspondent alors aux solutions recherchées. Donc, les pages faisant office d'autorités sur un sujet donné sont visibles (liées) depuis beaucoup de pages dans le sous-graphe étudié. Ainsi, en parcourant le graphe au hasard des liens, la probabilité de visiter une page autorité est grande.

Cette idée de détermination stochastique de l'intérêt d'une page se rapproche de celle utilisée par Brin et Page dans le calcul du PageRank mais diffère dans la séparation des résultats en hubs et autorités pour un domaine donné. La détermination des probabilités d'appartenance de chaque page du graphe à l'une des deux catégories possibles se fait à l'aide de chaînes de Markov.

La première étape consiste à construire un sous-graphe d'Internet correspondant à une requête donnée, sur lequel s'effectuera la recherche de pages pertinentes. Cette construction se base sur celle utilisée dans HITS et requiert les mêmes propriétés que celles décrites dans la section 6.2.

Afin de déterminer le potentiel de hub et d'autorité de chaque page, il faut déterminer la probabilité de visite de ces pages en suivant les liens du graphe de manière aléatoire. Le parcours des arcs dans le sens initial nous permet de déterminer les potentiels d'autorités des pages : on mesure ici la probabilité qu'une page soit pointée par beaucoup d'autres pages. Le parcours du graphe en suivant les arcs dans leur sens inverse nous donne les potentiels de hub de chaque page : on mesure, dans ce cas, la probabilité qu'une page pointe vers beaucoup d'autres pages.

Deux chaînes de Markov sont alors analysées : une chaîne de hubs et une chaîne d'autorités. Les états de transitions de ces chaînes sont générés en effectuant le parcours

aléatoire du graphe. Mais, contrairement à un parcours classique des liens, deux liens hypertextes sont traversés :

1) en choisissant selon une probabilité uniforme d'aller sur une page pointée par la page actuelle, et

2) en choisissant selon une probabilité uniforme d'aller sur une page pointant vers la page actuelle.

Les potentiels d'autorités sont alors définis comme étant la distribution stationnaire de la chaîne de Markov effectuant d'abord un pas 1) puis un pas 2), tandis que les potentiels de hubs sont définis comme la distribution stationnaire de la chaîne effectuant d'abord un pas 2) puis un pas 1).

Formellement, les deux matrices de transition des deux chaînes de Markov sont définies ainsi :

1. La matrice déterminant les potentiels de hub $\tilde{H}$:

$$\tilde{h}_{i,j} = \sum_{k:k\in S(i)\cap S(j)} \frac{1}{|S(i)|} \times \frac{1}{|E(k)|} \quad (11)$$

Equation II. 9 : Les potentiels de Hub.

2. La matrice déterminant les potentiels d'autorité $\tilde{A}$:

$$\tilde{a} = \sum_{k:k\in E(i)\cap E(j)} \frac{1}{|E(i)|} \times \frac{1}{|E(k)|} \quad (12)$$

Equation II. 10 : Les potentiels d'Autorité.

Dans ces formules, *E(i)* décrit tous les nœuds du graphe pointant vers le nœud *i*, et donc les pages que nous pouvons atteindre depuis la page *i* en suivant un lien dans le sens inverse. *S(i)* décrit tous les nœuds que nous pouvons atteindre depuis le nœud *i* en suivant un lien de *i*.

Une probabilité de transition $\tilde{a}_{i,j} > 0$ indique qu'une certaine page *k* pointe à la fois vers les pages *i* et *j* et que, de plus, la page *j* peut être atteinte depuis la page *i* en deux pas : en parcourant le lien de la page *k* vers *i* dans le sens inverse et en suivant ensuite le lien de la page *k* vers la page *j*.

La distribution stationnaire $h = (h_1, h_2, ..., h_N)$ de la chaîne de Markov satisfait $h_i = \frac{|S(i)|}{|S|}$ où $S = \cup_i S(i)$ est l'ensemble des liens sortants ; et que la distribution stationnaire

$a = (a_1, a_2, ..., a_N)$ de la chaîne de Markov satisfait $a_i = \frac{|E(i)|}{|E|}$ où $E = \cup_i E(i)$ est l'ensemble des liens entrants.

Cet algorithme n'établit pas les potentiels de *hub* et d'*autorité* de la même manière que l'algorithme HITS dans lequel la structure se renforce mutuellement. En effet, puisque, $a_i = \frac{|E(i)|}{|E|}$, l'*autorité* relative du nœud *i* est déterminée non pas depuis des liens globaux, mais depuis les liens locaux à ce nœud.

Le renforcement mutuel par la structure des liens du graphe de l'algorithme HITS fait qu'il pose problème dans certains cas, notamment ceux identifiés par l'effet TKC (Tightly-Knit Community - La communauté très unie). Cet effet apparaît lorsqu'une communauté de pages obtient un très bon score par les algorithmes d'établissement de pertinence par mesures topologiques, bien qu'elles ne fassent pas autorité sur le sujet donné. Ces communautés sont petites mais très fortement interconnectées entre elles. Les auteurs ont toutefois prouvé que bien que l'algorithme HITS soit affecté par ce problème, SALSA arrive à bien évaluer ces communautés.

7. Conclusion

Tout au de chapitre, nous avons défini le crawling et nous avons présenté les mécanismes de base nécessaire pour montrer la façon avec laquelle fonctionne le processus du crawling. Plus encore nous avons présenté les objectifs du crawling qui se résument en la détection et l'étude après détection des captures afin de déceler les propriétés largement étudiées des systèmes large échelle. Les problèmes qui confrontent la mise en œuvre d'un tel programme résident essentiellement dans la taille et la difficulté de communication entre les composants du programme ainsi qu'au refus de certains serveurs de répondre aux requêtes de crawling. Nous avons aussi présenté les crawlers les plus reconnus dans ce domaine [77].

Dans les chapitres qui suivent, nous présenterons notre crawler ainsi que les étapes que nous avons entreprises pour devenir indépendants des universités internationales en matière d'acquisition de l'information.

Chapitre III : État de l'Art sur la Recherche de l'Information

1. Introduction

Depuis les années 1940, le problème du stockage de l'information et de la Recherche d'Information a attiré une attention croissante. Nous avons de grandes quantités de données dont l'accès précis et rapide est en train de devenir de plus en plus difficile. Un effet de ceci, est que l'information pertinente sera ignorée, car elle n'est jamais découverte, ce qui conduit à beaucoup de duplication de travail et d'effort.

2. Définition

La Recherche d'Information (IR, une abréviation de Information Retrieval) est un terme large, souvent vaguement défini, mais dans ce travail, nous ne portons que de l'information automatique des systèmes de recherche. « La Recherche d'Information est le terme classique, bien que quelque peu inexacte. Un système de Recherche d'Information n'informe pas et ne change pas la connaissance de l'utilisateur sur l'objet de sa requête, il informe sur la seule existence ou la non-existence ainsi la localisation des documents relatifs à sa demande », une définition parfaitement claire donnée par Lancaster [78]. Cela exclut spécifiquement les systèmes de question-réponse comme illustrés par Winogradsky [79] et ceux décrits par Minsky [80]. Et exclut également les systèmes de Recherche de Données, tels que les systèmes utilisés par la Bourse de Devis en ligne.

L'introduction à la recherche d'information est le premier manuel à un traitement cohérent de la recherche classique, y compris la recherche Web et les domaines connexes de la classification de texte et regroupement de textes rédigés à partir d'un point de vue informatique [81]. Elle donne un traitement mis à jour de tous les aspects de la conception et la mise en œuvre des systèmes de collecte, d'indexation et la recherche de documents et de méthodes d'évaluation des systèmes, ainsi qu'une introduction à l'utilisation des méthodes d'apprentissage de collections de textes.

Nous utilisons le mot «document» comme un terme général qui pourrait également inclure des informations non textuelles, telles que des objets multimédias.

3. Naissance de la Recherche d'Information

Avec l'avènement des ordinateurs, une grande partie de la pensée a été donnée afin de l'utiliser pour fournir des systèmes de Recherche d'Information rapides et intelligents. Dans les bibliothèques, dont beaucoup ont certainement un stockage de l'information et de problème de Recherche d'Information, certaines des tâches les plus banales, comme le

catalogage et l'administration générale, ont été repris avec succès par les ordinateurs. Cependant, le problème de la recherche efficace reste largement non résolu.

Supposant que dans une bibliothèque de l'université il existe des milliers de livres, de mémoires et de thèses ; lorsqu'un étudiant ou un enseignant (un utilisateur) désire retirer ou emprunter un livre sur un certain sujet (une requête), le bibliothécaire va sûrement lui apporter plusieurs références de livres satisfaisants qui évoquent ce sujet (variation de réponses) et supposant que cet utilisateur n'a pas déjà une idée à-propos des éditeurs ou des titres de ces derniers livres.

Il va se poser la question suivante : quel-est le livre qui correspond exactement à ce que je cherche ? Peut-être même qu'il va poser cette question au bibliothécaire s'il a déjà une expérience à propos de ce sujet ! Sinon, il commence à feuilleter ces livres dans la bibliothèque, en conservant les livres pertinents et à éliminer tous les autres. Ce qui constitue une Recherche Parfaite.

Cette solution est évidemment impossible. Un utilisateur n'a pas le temps ou ne souhaite pas de passer le temps à lire toute la collection de livres, mis à part le fait qu'il peut être physiquement impossible pour lui de le faire.

A l'arrivée des ordinateurs à haute vitesse pour les tâches non-numériques (en langage naturel), beaucoup pensaient qu'il serait possible de lire des collections de documents afin d'en extraire un ensemble de documents pertinents. Il est vite devenu évident que l'utilisation du texte en langage naturel d'un document cause non seulement des problèmes d'entrée et de stockage, mais aussi laisse en suspens le problème intellectuel de la caractérisation du contenu du document.

Il est concevable que les développements futurs matériels puissent faire l'entrée et le stockage du langage naturel plus possible. Mais la caractérisation automatique qui essaye de générer le processus de « Lecture » humain est en effet un problème très délicat. La «Lecture» implique une extraction, à la fois syntaxique et sémantique du texte et de l'utiliser pour décider si chaque document est pertinent ou non à une demande particulière.

La difficulté n'est pas seulement de savoir comment extraire les informations mais aussi comment les utiliser pour décider de la pertinence. Le progrès relativement lent de la linguistique moderne sur le front sémantique et l'échec apparent de la traduction automatique montrent que ces problèmes sont en grande partie non résolus [82].

Donc la notion de la «*Pertinence*» c'est au centre de la recherche d'information. Alors le but d'une meilleure stratégie de Recherche d'Information automatique est de rechercher tous les documents pertinents tout en minimisant la recherche et la récupération des documents non-pertinents que possible.

Lorsque la caractérisation d'un document est élaborée, elle doit lui permettre qu'il soit présent dans la réponse s'il est pertinent à cette requête sans changer sa représentation. Des indexeurs humains ont toujours caractérisé les documents de cette manière lors de l'attribution des termes d'indexation des documents. L'indexeur tente d'anticiper le genre de termes d'indexation qu'un utilisateur va utiliser afin de rechercher chaque document dont le contenu

qu'il est entrain de décrire. Implicitement, il est entrain de construire des requêtes pour lesquelles le document est pertinent.

Lorsque l'indexation est automatiquement faite, il est supposé qu'en développant le texte d'un document ou d'une requête avec le même analyseur automatique, la sortie sera une représentation du contenu, et si le document est pertinent à la requête, il sera montré dans des statistiques.

Il est possible pour un être humain de montrer la pertinence d'un document à une requête. Pour un ordinateur, pour faire, nous avons besoin de construire un modèle dans lequel les décisions de pertinence peuvent être quantifiées.

4. Un model typique d'un système de Recherche d'Information

Le but des systèmes de Recherche d'Information est généralement de trouver l'information dans des textes et des documents (par exemple [w3, w4, w5, w6, w7]).

Les modèles de systèmes de Recherche d'Information sont généralement représentés sous la forme décrite dans la figure III.1 [83], avec des quantités variables de détails descriptifs supplémentaires en fonction de l'objet de la description.

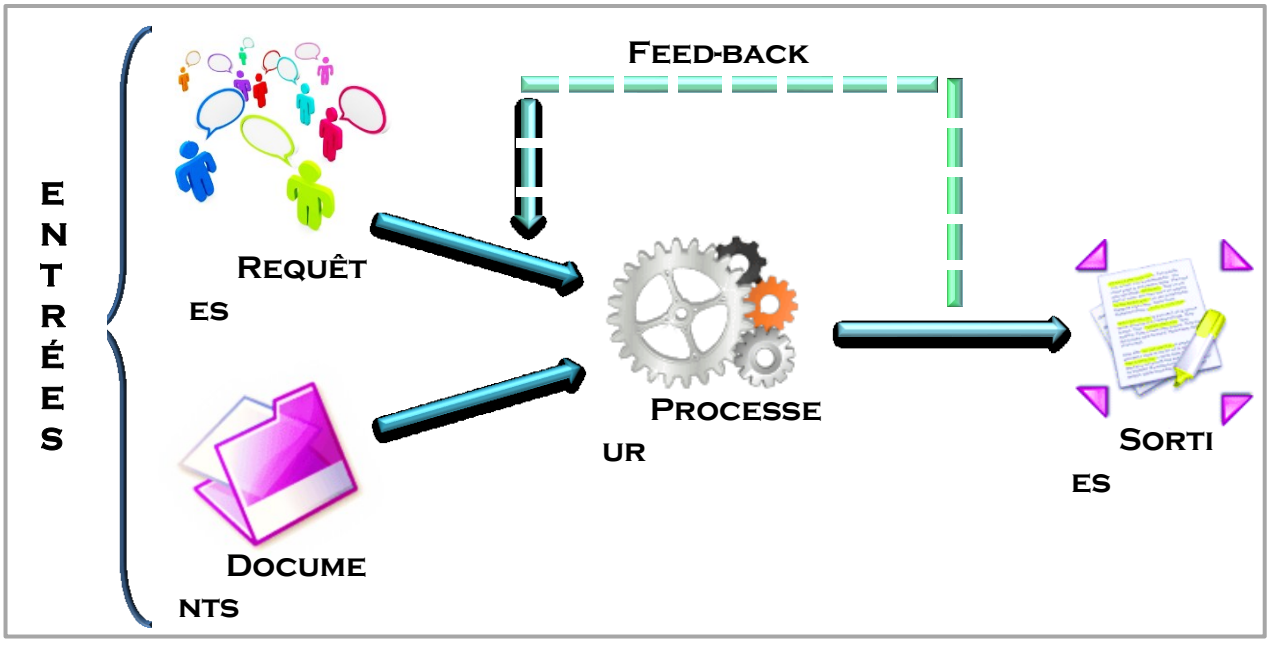


Figure III. 1 : Model de systèmes de Recherche d'Information.

Tague et al. [w8] ont publié un « *modèle formel complet pour les systèmes de recherche* » qui utilise une production grammaticale et des hypergraphes pour représenter la structure du texte, l'indexation et l'accès. Toutefois, ce n'est qu'un modèle procédural de techniques d'extraction de texte. Les descriptions du fonctionnement des systèmes de recherche individuelles sont susceptibles d'avoir des organigrammes détaillés des composants de ce

Clicours.com

système particulier. Ici, cependant, nous sommes intéressés à développer une analyse fonctionnelle complète et générale des systèmes de Recherche d'Information.

La figure montre trois composantes : L'entrée, la sortie et le processeur. En commençant par les entrées ; Le principal problème ici est d'obtenir une représentation de chaque document convenable à une requête donnée afin qu'elle soit utilisée par un système.

La plupart des systèmes de Recherche d'Informatiques stockent seulement une représentation du document (ou requête) ce qui signifie que le texte d'un document est perdu une fois qu'il a été traité dans le but de générer sa représentation. Un document représentant pourrait devenir, par exemple, une liste de mots extraits jugés significatifs. Plutôt que d'avoir le processus en langage naturel, une approche alternative est d'avoir un langage artificiel dans lequel toutes les requêtes et les documents peuvent être formulées [84]. Bien sûr, l'utilisateur est prêt à exprimer son besoin d'information dans la langue.

Lorsque les systèmes de recherche sont en ligne, il est possible pour un utilisateur de modifier sa demande au cours d'une session de recherche, ce qui améliore par la suite l'exécution de la recherche. Une telle procédure est couramment appelée rétroaction (feedback).

En second lieu, le processeur, la partie du système concerné par la procédure de la recherche. Il implique la structuration de l'information d'une manière quelconque appropriée, telle que la classification, indexation, les listes inversées, …etc. Il impliquera également de remplir la fonction réelle de la recherche, l'exécution de la stratégie de recherche afin de trouver une réponse à une requête.

Les documents ne sont pas seulement l'entrée mais ils peuvent être utilisés au cours du processus d'extraction de manière à ce que leur structure soit plus correcte, considérée comme faisant partie du processus de recherche.

Enfin, nous arrivons à la sortie, qui est généralement un ensemble de citations ou des numéros de document. Le but de la recherche d'information (RI) est de fournir aux utilisateurs de ces documents qui sauront satisfaire leur besoin d'information.

Après avoir fourni un modèle général du processus de recherche d'information, nous décrivons ensuite brièvement les principales méthodes d'extraction de texte en fonction de leurs forces et faiblesses.

5. Principaux modèles d'extraction de texte :

Les grands modèles suivants ont été développés pour récupérer des informations : le modèle Booléen, le modèle statistique, qui comprend l'espace vectoriel et le modèle probabiliste de récupération et le modèle basé sur les connaissances et les linguistiques. Le premier modèle est souvent désigné comme le modèle qui a une «correspondance exacte » et les derniers comme une «meilleure correspondance » [w9].

Généralement les requêtes sont moins que parfaites puisqu'elles récupèrent des documents pertinents et au même temps elles ne récupèrent pas tous les documents pertinents. Ces deux

points sont utilisés pour évaluer l'efficacité d'un procédé d'extraction. Le premier, appelé le *taux de précision (precision rate)*, qui est égale à la proportion de documents extraits qui sont réellement pertinent. Le second, appelé le *taux de rappel (recall rate)*, qui est égale à la proportion de tous les documents pertinents qui sont effectivement récupérés. Si les chercheurs veulent augmenter la précision, alors ils doivent affiner leurs requêtes. Si les chercheurs veulent élever le taux de rappel, ils élargissent leur requête. En général, il existe une relation inverse entre la précision et le rappel. Les utilisateurs ont besoin d'aide pour se familiariser avec la façon de gérer le compromis entre la précision et le rappel pour leur besoin de l'information [w10].

5.1 Le modèle Booléen

5.1.1 Booléen Standard

	La Démarche Booléenne Standard
But	Structure conceptuelle et contextuelle de l'information
Méthodes	• Coordination : ET, OU, NON • contraintes de proximité • Champ, domaine • Stop Word
(+)	▪ Facile à implémenter ▪ Grande puissance et la clarté expressive ▪ Techniques pour élargir ou affiner une requête
(-)	○ Construction des requêtes booléennes difficile ○ Tout ou rien ○ ET trop sévère, et OU ne fait beaucoup de différence

Tableau III. 1: La démarche booléenne standard.

Dans le tableau III.1, nous trouvons les caractéristiques essentielles de la démarche Booléenne et la liste de ses principaux avantages et inconvénients. Il dispose des atouts suivants :

- Il est facile à mettre en œuvre et il est informatiquement efficace [w11]. Par conséquent, il est un modèle standard pour les systèmes de recherche opérationnelle et la plupart des grands services d'information en ligne l'utilisent.
- Il permet aux utilisateurs d'exprimer des contraintes structurelles et conceptuelles pour décrire les caractéristiques linguistiques importantes [w10]. Les utilisateurs trouvent que des synonymes de spécifications (réfléchis par OU-clauses) et des

phrases (représentées par des relations de proximité) sont utiles dans la formulation de requêtes.

- L'approche booléenne possède une grande puissance et une clarté expressive. La recherche booléenne est très efficace si une requête nécessite une sélection exhaustive et sans ambigüité.
- La méthode booléenne offre une multitude de techniques pour élargir ou affiner une requête.
- L'approche booléenne peut être particulièrement efficace dans les étapes ultérieures du processus de recherche, en raison de la clarté et la rigueur avec laquelle les relations entre les concepts peuvent être représentées.

L'approche booléenne standard a les défauts suivants :

- Les utilisateurs trouvent qu'il est difficile de construire des requêtes booléennes efficaces pour plusieurs raisons : les utilisateurs utilisent les termes en langage naturel ET, OU ou NON qui ont un sens différent lorsqu'ils sont utilisés dans une requête. Ainsi, les utilisateurs pourront faire des erreurs quand ils forment une requête booléenne, car ils ont recours à leurs connaissances de l'anglais [w12, w13, w9].

5.1.2 Méthode booléenne intelligente

Il y a eu des tentatives pour aider les utilisateurs à surmonter certains des inconvénients de l'opération booléenne traditionnelle discuté ci-dessus. Nous allons maintenant décrire un tel procédé, appelé intelligent booléenne, développé par Marcus [w10, w14] qui cherche à aider les utilisateurs à construire et modifier une requête booléenne ainsi que faire de meilleurs choix sur les dimensions d'une requête booléenne. Nous ne tentons pas de fournir une description détaillée de la méthode booléenne intelligente, mais de l'utiliser comme un bon exemple qui illustre certains des moyens possibles pour faire la récupération Booléenne plus efficace et convivial.

Les utilisateurs commencent en spécifiant une déclaration en langage naturel qui est automatiquement traduite en une représentation booléenne du sujet qui consiste en une liste de facteurs ou de concepts, qui sont automatiquement coordonné avec l'opérateur ET. Si l'utilisateur à l'étape initiale peut ou veut inclure des synonymes, ils sont coordonnés en utilisant l'opérateur OU. Par conséquent, la représentation du sujet Booléen relie les différents facteurs à l'aide de l'opérateur ET, où les facteurs peuvent consister en des termes simples ou plusieurs synonymes reliés par l'opérateur OU.

L'un des objectifs de l'approche intelligente booléenne est de faire usage de la connaissance structurelle contenue dans les substituts de texte, où les différents domaines représentent des contextes d'informations utiles. En outre, l'approche intelligente Booléenne peut faire ressortir une racine (radical) commune entre ces différents concepts. Par exemple, les concepts «information» et «informatique» ont la racine commune *informat**.

La stratégie initiale de l'approche booléenne intelligente est de commencer avec la requête la plus large possible dans les limites de la façon dont les facteurs et de leurs synonymes ont été coordonnées. Par conséquent, il modifie la représentation du sujet booléen dans la substitution de la requête en utilisant uniquement les racines des concepts et il lance la recherche à travers le système. Une fois que la substitution de la requête a été effectuée, les utilisateurs sont guidés dans le processus d'évaluation des substituts de documents récupérés. Ils choisissent parmi une liste de raisons pour indiquer pourquoi ils estiment certains documents pertinents. De même, ils peuvent indiquer pourquoi les autres documents ne sont pas pertinents en interagissant avec une liste de raisons possibles. Ce retour d'utilisateur est utilisé par le système booléen intelligent pour modifier automatiquement la représentation booléenne des sujets ou la substitution de la requête, ce qui est plus approprié.

L'approche Booléenne intelligente offre un riche ensemble de stratégies pour modifier une requête basée sur le retour de pertinence reçu ou le besoin exprimé pour affiner ou élargir la requête. Le paradigme de la récupération booléenne intelligent a été mis en œuvre sous la forme d'un système appelé CONIT, qui est l'un des premiers systèmes de recherche d'experts qui a été en mesure de démontrer que les utilisateurs ordinaires, assistés par un tel système, pourrait effectuer aussi bien en tant qu'intermédiaires de recherche connu [w14]. Toutefois, les utilisateurs doivent naviguer à travers une série de Pages menus des choix différents, où il pourrait être difficile pour eux d'apprécier les conséquences de certains de ces choix.

5.2 Modèle statistique

L'espace des modèles *vectoriels* et *probabilistes* sont les deux principaux exemples de l'approche de recherche statistique. Les deux modèles utilisent des informations statistiques sous forme de fréquences à long terme afin de déterminer la pertinence des documents par rapport à une requête. Bien qu'ils diffèrent dans la manière dont ils utilisent les fréquences terme, les deux, produisent leur sortie sous forme d'une liste de documents classés selon leur pertinence estimée. Les modèles de recherche statistiques résoudront certains des problèmes de méthodes de recherche booléennes, mais ils ont des inconvénients. Nous décrirons également *l'indexation sémantique latente* et *regroupement* des approches qui sont basées sur des approches d'extraction de statistiques, mais leur objectif est de répondre à ce que la requête de l'utilisateur n'a pas dit, ne pouvait pas dire, mais de toute façon manifeste [w15, w16].

5.2.1 Modèles des espaces vectoriels

Le **modèle d'espace vectoriel** représente les documents et les requêtes en tant que vecteurs dans un espace multidimensionnel, dont les dimensions sont les termes utilisés pour construire un indice afin de représenter les documents [w17]. La création d'un index implique une analyse lexicale pour identifier les termes importants, où l'analyse morphologique réduit les différentes formes de mots de commune "racines", et l'apparition de ces racines est calculée. Les vecteurs de requête et de document sont comparés en utilisant, par exemple, la mesure de similarité cosinus. Dans ce modèle, les conditions d'une substitution de la requête

peuvent être pondérées pour tenir compte de leur importance, et ils sont calculés en utilisant les distributions statistiques des termes dans la collecte et dans les documents [w17]. Le modèle d'espace vectoriel peut attribuer une cote de haut rang à un document qui ne contient que quelques-uns des termes de la requête si ces conditions se produisent rarement dans la collection, mais souvent dans le document. Le modèle d'espace vectoriel repose sur les hypothèses suivantes :

1) Plus un vecteur de document est proche d'un vecteur de la requête, plus il est probable que ce document est pertinent pour cette requête.

2) Les termes utilisés pour définir les dimensions de l'espace sont orthogonaux ou indépendants. Tant qu'il y a une première approximation raisonnable, l'hypothèse que les mots qui sont indépendants deux à deux n'est pas réaliste.

5.2.2 Modèle probabiliste

Le **modèle probabiliste** est basé sur le principe classement selon la probabilité, qui dispose qu'un système de recherche de l'information est censée de classer les documents en fonction de leur probabilité de pertinence à la requête, étant donné toutes les preuves disponibles [w9]. Le principe tient compte qu'il existe une incertitude dans la représentation de l'information et des documents. Il peut y avoir une variété de sources de données qui sont utilisées par les méthodes d'extraction probabilistes, et le plus commun est la distribution statistique des termes dans deux documents pertinents et non pertinents.

Les approches statistiques ont les atouts suivants :

- Ils offrent aux utilisateurs un classement de pertinence des documents trouvés. Par conséquent, ils permettent à l'utilisateur de contrôler la sortie en fixant un seuil de pertinence ou en spécifiant un certain nombre de documents à afficher.
- Les requêtes peuvent être plus faciles à formuler, car les utilisateurs n'ont pas à apprendre un langage de requête puisqu'ils peuvent utiliser un langage naturel.
- L'incertitude inhérente dans le choix des concepts de la requête peut être représentée.

Cependant, les approches statistiques ont les lacunes suivantes :

- Ils ont un pouvoir expressif limité. Par exemple, l'opération ne peut pas être représentée parce que seuls les poids positifs sont utilisés. Par exemple, la requête booléenne très fréquente et importante ((A et B) ou (C et D)) ne peut pas être représenté par une requête d'espace vectoriel. Par conséquent, les approches statistiques n'ont pas la puissance expressive de l'approche booléenne.
- L'approche statistique n'a pas une structure pour exprimer les caractéristiques linguistiques importantes telles que des phrases, les contraintes de proximité sont également difficiles à exprimer, une fonction qui est d'une grande utilité pour les chercheurs expérimentés.
- Le calcul des scores de pertinence peut être coûteux en calcul.

- Les requêtes doivent contenir un grand nombre de mots en vue d'améliorer le rendement de recherche. Comme c'est le cas pour l'approche booléenne, les utilisateurs sont confrontés au problème d'avoir à choisir les mots appropriés qui sont également utilisés dans les documents pertinents.

5.3 Approches basées sur les connaissances et linguistiques

Dans la forme la plus simple de la recherche automatique de texte, les utilisateurs entrent une chaîne de mots-clés qui sont utilisés dans la recherche des indices inversés des documents. Cette approche récupère les documents basés uniquement sur la présence ou l'absence d'une chaîne exacte d'un seul mot, spécifié par la représentation logique de la requête. Il est clair que cette approche va manquer beaucoup de documents pertinents, car il ne saisit pas le sens complet ou profond de la requête de l'utilisateur. Les approches basées sur les connaissances et linguistiques ont été également développées pour résoudre ce problème en effectuant une analyse morphologique, syntaxique et sémantique afin de rechercher des documents plus efficacement [w3].

Dans une analyse morphologique, les racines et les affixes sont analysés pour déterminer la partie des mots du discours (nom, verbe, adjectif, etc.). Les phrases complètes qui suivant doivent être analysées (parsées) en utilisant une certaine forme d'analyse syntaxique. Enfin, les méthodes linguistiques doivent résoudre les ambigüités de texte et / ou de générer des synonymes pertinents ou quasi-synonymes basées sur les relations sémantiques entre les mots. Le développement d'un système de récupération linguistique sophistiquée est difficile et nécessite des bases de connaissances complexes de l'information sémantique et heuristiques pour la recherche. Par conséquent, ces systèmes nécessitent souvent des techniques qui sont communément appelées techniques des systèmes de renseignement ou experts artificiels.

Niveau linguistique	Recherche booléenne	Statistique	Bases de connaissances & linguistiques
Lexical	Liste de stop-mots	Liste de stop-mots	Vocabulaire
Morphologique	Troncature de symboles	Radical (racine)	Analyse morphologique
Syntaxique	Opérateurs adjacents	Phrases statiques	Phrases grammaticales
Sémantique	Dictionnaires	Clusters des cooccurrences des mots	Index de mots/phrases pour les relations sémantiques

Tableau III. 2 : Les caractéristiques des principales méthodes de recherche.

5.4 Conclusion sur les techniques d'extraction de texte

Il y a un décalage croissant entre l'approche de recherche utilisée par les systèmes de recherche commerciaux existants et les approches étudiées et promues par une grande partie de la communauté de recherche d'information. La première est basée sur le modèle de booléen ou la correspondance exacte, les autres sont les approches statistiques et linguistiques, aussi appelés les approches de correspondance partielle. Tout d'abord, la principale critique formulée contre l'approche booléenne est que ses requêtes sont difficiles à formuler. Deuxièmement, l'approche booléenne permet de représenter des informations structurelles et contextuelles qui seraient très difficile à représenter à l'aide des méthodes statistiques. Troisièmement, les approches de la correspondance partielle fournissent aux utilisateurs une sortie classée, mais ces listes de classement cachent des informations précieuses. Quatrièmement, les expériences de recherche récentes ont montré que les approches de correspondance exacte et partielles sont complémentaires et doivent donc être combinées [w18].

Dans le tableau III.3, nous résumons quelques-uns des principaux problèmes dans le domaine de la recherche d'information ainsi des solutions possibles.

Problèmes	Solutions Possibles
Sélection de la recherche lexicale	• Dictionnaires • Indexation sémantique cachée
Reformulation de la stratégie de recherche	• Utilisation des booléens intelligents • Approches linguistiques et statistiques • Dictionnaires • Interfaces graphiques
Surcharge des informations	• Classement • Clusterisation • visualisation

Tableau III. 3 : Problèmes dans le la recherche d'information et les solutions possibles.

6. Conclusion

Dans un monde idéal, les trois processus (Représentation, Index décisions et l'élaboration de requêtes) s'appuient tous sur des sources de connaissances identiques, mais il est peu probable dans la pratique. La Recherche d'Information n'informe pas et ne change pas la connaissance de l'utilisateur, par contre c'est un outil qui aide l'utilisateur de capter facilement et rapidement ses requêtes et ses connaissances. Nous avons évoqué quelques modèles en décrivant leurs avantages et inconvénients.

Chapitre IV : Conception

1. Introduction

Notre travail traite de l'exploration des pages Web d'un côté et d'un autre, la fouille de textes dans le cadre d'une application au domaine de la recherche d'information (IR). Il s'agit de concevoir un crawler performant du Web qui est un processus de collecte et d'analyse des pages Web. Il permet le parcours des pages Web afin de traiter plusieurs points, à savoir : la détection des propriétés des graphes du Web, les différentes formes de changement qui surviennent et affectent le World Wide Web ainsi que l'indexation des données telles que les images, les vidéos, les documents, …etc. Nous traitons aussi l'extraction du texte afin de créer un index de termes avec leurs liens hypertextes ainsi que leurs nombres d'apparitions.

Ce problème est réel et difficile, il nous porte vers un défi d'un point de vue Crawling, où nous étions toujours dépendants des autres universités afin qu'ils puissent nous fournir des captures et qui sont généralement déjà étudiées, alors que nous sommes dans le besoin d'avoir des captures vierges à explorer. D'un point de vue de Recherche d'Information, le développement d'un tel système sophistiqué est difficile et nécessite des bases de connaissances complexes de l'information sémantique et heuristique pour la recherche. Par conséquent, ces systèmes nécessitent souvent des techniques d'experts artificiels.

2. Objectifs

Notre objectif est de concevoir et réaliser un robot d'indexation performant qui explore le web d'une manière automatique, méthodique et ordonnée afin de pouvoir créer un résumé textuel du contenu des pages Web, tels que les adresses URLs, des éléments graphiques, textes, vidéos , …etc. Le robot nécessite une plateforme complémentaire basé sur l'HTML qui suit la structure basique d'une page Web.

La collecte générée depuis un Crawler, fournit des captures du réseau sciée dans une période de temps afin de déterminer et de caractériser les connexions établies par les liens hypertextes des pages Web en modélisant la topologie d'Internet sous forme de graphe.

L'interrogation de la topologie de connexion des pages Web entre elles amène des informations très importantes sur la manière de rechercher de l'information dans le Web. La vaste distribution des pages Web à travers le monde entier oriente vers la manière dont les pages sont-elles reliées entre elles et comment les atteindre toutes.

D'ici nous avons identifié quelques défis principaux afin de répondre aux problématiques de cette thèse :

1. Implémenter un Crawler performant qui répond aux critères théoriques.
2. Le passage à l'échelle dans la collecte des ensembles d'informations.

3. Crawler une nouvelle zone dans le monde, une nouvelle contribution dans les recherches.
4. Visiter la partie Arabe et Algérienne du Web.
5. Procéder à la création d'un système de Recherche d'information.

Dans ce chapitre, nous allons premièrement faire des études sur les fichiers capturés afin d'en déterminer les points suivants :

- Calcul du diamètre global ;
- Calcul du coefficient d'agrégation ou de clusterisation ;
- Calcul de la composante géante, qui représente la composante fortement connexe dans un graphe orienté ;
- Calcul du coefficient de la loi de puissance λ ;
- La distribution des degrés entrants et sortants des pages Web ;
- La distribution des degrés dans un graphe non-orienté ;
- Diamètre du graphe ;
- La profondeur maximale ;
- Nombre de cycles ;
- La distance moyenne.

Afin de foisonner ce travail, un module qui traite l'extraction du texte est ajouté. Notre objectif est de répondre le mieux aux exigences des profils des utilisateurs, pour cela nous avons conçu une plateforme qui extrait le texte à partir des pages Web, non seulement en lettres latines, mais aussi en lettres arabes. C'est une première ainsi une recherche qui récupère le texte à partir des documents en extension PDF.

Afin de garantir nos buts et objectifs, une étude spécifique et détaillée est faite à propos du protocole HTTP (Hyper Text Transfer Protocol) afin de le bien maitriser pour son implémentation et pour faire un bon traitement des informations des pages Web.

3. Le Protocole HTTP et le langage HTML

Le protocole HTTP est un protocole de communication pour le transfert de documents. La communication se fait entre un client (machine envoyant des requêtes) et un serveur (machine répondant à ces requêtes) développé pour le World Wide Web. Ce protocole est utilisé par les serveurs Web hébergeant des sites Internet, dans le but de permettre de télécharger des documents ainsi que la consultation de pages sur l'écran du client.

Afin de communiquer avec un serveur Web sans passer par un navigateur, il faut connaitre le http pour atteindre une automatisation exigée par les robots d'exploration du Web.

Dans notre application, nous avons fourni une liste des liens où à partir de ces liens, le robot initie son exploration. Cette liste est appelée les graines. Notre Crawler commence sa visite lien par lien et à chaque nouveau lien rencontré, le Crawler fait une identification de tous les liens hypertexte dans cette page afin qu'il puisse les ajouter dans la liste de la frontière Index. Ces derniers sont récursivement visités ensuivant la politique de la profondeur d'abord.

Pour cela, il faut localiser tout lien hypertexte dans les pages Web à visiter dans les graines. La mise en forme des pages Web est représentée par un langage nommé HTML. Dans la suite nous allons décrire ce type de langage, sa structure générale et comment peut-on extraire les données contenues dans ce genre de documents (Plus de détails dans la section suivante).

HTML est un langage universel utilisé pour communiquer sur le Web. L'information sera ainsi transportée sur cette gigantesque toile Internet, en d'autres termes, HTML est un ensemble (réduit) de balises (ou styles ou "tags") utilisés pour définir les différents composants d'un document. Nous donnons l'aspect général de ce langage :

3.1 Structure générale d'un document HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
   <html>
      <body>
           <p>Liste de nombres :</p>
                <ul>
                     <li>10 : dix</li>
                     <li>0 : zéro</li>
                     <li>33 : trente-trois</li>
                     <li>7 : le premier nombre parfait</li>
                </ul>
   </body>
   </html>
```

Nous nous intéressons ici à la balise A et/ou Link suivante car elle contient les liens hyperliens vers les contenus dans les pages Web. Généralement, la balise est du format :

```
<A HREF="url">un-petit-texte</A>.
```

La balise `<A></A>` peut représenter différents types de liens comme image, fichier, mail… etc. le paragraphe suivant illustre quelques exemples de la balise `<A></A>`.

Clicours.COM

3.2 Quelques types des hyperliens

• Un autre document HTML :

```
<A HREF="http://www.urec.fr/index.html">Serveur Web de l'UREC</A>
```

• Un lien vers une image :

```
<A HREF="retina.jpg">Image de la rétine</A>
```

• Un lien sur fichier son, ou vidéo à charger, un fichier compressé :

```
<A HREF="anim.mov">cliquez ici pour visionner l'animation</A>
```

• Un lien vers un programme de composition de mail :

```
<A HREF="mailto:adresse"> texte </A>
```

• Cette directive mailto permet de spécifier le champ sujet :

```
<A HREF="mailto:adresse?subject=Sujet"> texte</A>
```

3.3 Extraction de Liens d'une page HTML

Afin de pouvoir lister les liens d'une page Web, il faut tout d'abord qu'il y a une connexion ouverte, après le chargement des pages HTML avec *URLConnection* on aura la possibilité de lire les pages une après l'autre avec *HTMLEditorKit* en parcourant les balises `href` pour extraire les liens.

Les trois étapes sont illustrées par les codes suivants :

```
URL url = new URL("http://www.univ-oran.dz");
        URLConnection uconnection = url.openConnection();
        Reader rd = new InputStreamReader(uconnection.getInputStream());
```

Code IV. 1 : Chargement d'une page Web.

```
EditorKit kit = new HTMLEditorKit();
        HTMLDocument doc = (HTMLDocument) kit.createDefaultDocument();
        doc.putProperty("IgnoreCharsetDirective", new Boolean(true));
        kit.read(rd, doc, 0);
```

Code IV. 2 : Lecture du document HTML.

```
HTMLDocument.Iterator it = doc.getIterator(HTML.Tag.A);
        while (it.isValid()) {
            SimpleAttributeSet s = (SimpleAttributeSet) it.getAttributes();
            String link = (String) s.getAttribute(HTML.Attribute.HREF);
            if (link != null) {
                //Afficher le lien trouvé
                lien.add(link);
            }
            it.next();
        }
```

Code IV. 3 : Parcours extraction des liens.

3.4 Extraction de Texte d'une page HTML

Le langage HTML est très similaire à XML, on pourrait donc être tenté de l'analyser à l'aide d'une API dédiée à XML (par exemple XmlReader, XmlDocument ou Linq to XML), et dans certains cas cela fonctionnerait.

Malheureusement, si on fait exception du cas particulier de XHTML, la plupart des pages HTML ne sont pas des documents XML valides, car HTML est beaucoup moins strict que XML. Un parseur XML normal ne pourra donc pas analyser une page HTML, car il échouera à la première erreur rencontrée (par exemple un attribut sans guillemets ou un élément non refermé).

import org.htmlparser.Parser ; c'est une bibliothèque open-source qui permet d'analyser le contenu d'une page web via le DOM (Document Object Model), avec une API (Application Programming Interface) similaire à Linq to XML, et qui permet aussi l'utilisation de Xpath.

Une fois la connexion est ouverte, le chargement des documents html débute au même temps et le parseur commence à parcourir l'arborescence du document. Le DOM est constitué de « nœuds » imbriqués, qui représentent des éléments (ou balises) HTML, du texte, ou des commentaires.

Le nœud racine représente le document lui-même ; on y accède via la propriété *parser.visitParentNode*. A partir de cette racine, on va « descendre » dans le document pour trouver les nœuds qui nous intéressent. Afin de récupérer le contenu entre les balises de ces nœuds, on peut utiliser la propriété suivante :

parser.setInputHTML(html)

parser.visitAllNodesWith (html.getStrings)

3.4.1 Extraction du texte en langue Arabe

Les trois attributs qui permettent l'internationalisation du HTML sont :

- « lang » (language) ;
- « dir » (direction) ;

- « charset » (character set).

Ces trois attributs prennent une importance considérable quand il s'agit de développer des sites en arabe. En effet, si par exemple l'attribut « dir » n'est pas utilisé dans les sites qui utilisent des langues s'écrivant de gauche à droite (cette direction étant par défaut), il est en revanche indispensable pour les sites en arabe (et toute autre langue qui s'écrit de droite à gauche comme le persan ou l'hébreu).

- ***« lang »***

L'attribut s'écrit : *lang = "language-code"*, l'attribut définit la langue de base des valeurs des attributs d'un élément et le texte que cet élément contient. Dans le cas de l'arabe l'attribut s'écrit lang="ar".

Il faut faire la différence entre deux situations d'utilisation de l'attribut :

- Si la langue primaire de la page (ou la langue par défaut) est l'arabe, l'attribut est ajouté à la balise `<html>` du document : `<html lang="ar">` ;
- Si la langue primaire de la page n'est pas l'arabe, l'attribut s'ajoute à la balise qui définit le contenu arabe, par exemple : `<p lang="ar">`.

- ***« dir »***

L'attribut s'écrit : `dir = "ltr | rtl"` (left to right- valeur par défaut - ou right to left). L'attribut définit la direction de base d'un texte à direction neutre (c'est-à-dire un texte qui n'a pas de définition de direction inhérente comme définie par UNICODE) dans le contenu d'un élément.

Cet attribut permet aux langues qui s'écrivent de droite à gauche comme l'arabe d'apparaître dans le navigateur effectivement de droite à gauche.

Comme pour l'attribut « lang », il existe deux cas de figure :

- La direction par défaut de la page est de droite à gauche, et l'attribut s'ajoute à la balise `<html>` : `<html dir="rtl">`.
- La direction de la page est de gauche à droite, et l'attribut s'ajoute aux valeurs des attributs de l'élément qui contient le texte de droite à gauche : `<p dir="rtl">`.

En général, et en ce qui concerne l'arabe, les deux attributs « lang » et « dir » sont utilisés de concert : `<html lang="ar" dir="rtl">` ou `<p lang="ar" dir="rtl">`.

- ***« Charset »***

Contrairement à d'autres langues, la langue arabe n'a pas eu beaucoup de chance avec l'informatique. Et bien qu'il y ait eu un standard *ISO* (ISO-8859-6) pour adresser le jeu de caractères arabes, pratiquement chaque compagnie de matériel ou de logiciel a « inventé » son jeu à elle (IBM, Microsoft, Apple, Linotype...). De tous ces jeux de caractères propriétaires il ne reste plus effectivement en utilisation que deux : Mac arabe et Windows arabe. Le résultat

est que si on produit un texte arabe sur Mac, ce texte n'est pas lisible sur Windows ou sur Unix-Linux, sauf dans le cas particulier où une application supporte les deux jeux de caractères même la version arabe de Microsoft Word a besoin d'une extension non fournie en standard pour lire les textes en Mac arabe. Bien que le standard ISO règlerait le problème, mais les applications qui sauvegardent le texte en arabe ISO se comptent sur les doigts d'une seule main.

Avec l'arrivée de *UNICODE* (UTF8), le problème semblait se résoudre car les deux systèmes (Mac et Windows) supportent ce standard. Tout cela influence évidemment le choix du codage des sites arabes. Si on s'en tient à une vision étroite et non professionnelle des choses, on sait que la grande majorité des utilisateurs arabes du web utilisent Internet Explorer sous Windows et donc que le plus simple est de choisir le codage Windows pour les textes arabes (Windows-1256).

Les navigateurs sous Windows supportent Windows-1256, ISO-8859-6 et UTF8. Mais les versions pour Mac ne supportent rien du tout, ce qui fait que si on veut visiter des sites arabes à partir d'un Mac il faut avoir Navigator 6 ou Mozilla. Quant à Opera, il supporte UTF8 et, théoriquement, ISO-8859-6, mais ces deux derniers formats ne marchent pas sur Mac, en plus Opera a tendance à ne pas supporter la direction de droite à gauche.

En conclusion, on doit être au courant de toutes ces informations qui sont nécessaires, car au moment de l'extraction du texte on parcoure le document HTML en cherchant les balises significatives qui aident à bien représenter les données afin de les stocker pour un traitement ultérieur. Finalement, la condition minimale pour l'extraction d'une page Web contenant du texte arabe est d'exploiter les trois attributs « lang », « dir » et « charset ».

4. Spécifications du Crawler

4.1 La capture

La première chose que nous avons faite est de lancer notre crawler dans les réseaux web à travers d'un cache initiale qui contient un ensemble de pages de démarrage, les graines. Ensuite le crawler commence à se faufiler dans le réseau et capture toutes les pages web possibles et chaque page subit une extraction des liens HyperText et d'objets possibles jpg, empg, mp4,…, etc.

Après avoir extrait les liens, on sauvegarde l'adresse de la page dans un fichier texte : captur.txt et la trace de la capture dans un autre fichier nommé : Index.cwf.

Dans le fichier Index.cwf, la première ligne correspond à la date début de la capture et la dernière indique la date fin de cette capture. Les autres lignes contiennent les successeurs de chaque page Web.

Dans le fichier captur.txt. Nous avons codifié les pages par un entier, cette codification permet d'alléger la taille du fichier Index.cwf. Plus de détails sont donnés dans la partie suivante.

4.2 Choix du parcours de la toile du Web

Nous avons choisi la stratégie de parcours en profondeur d'abord d'où on visite le graphe du Web en partant d'un sommet (première page des graines), on cherche le plus long chemin en suivant les fils de la dernière page visitée à gauche ou à droite puis on revient en arrière pour suivre les autres chemins possibles. Les graphes générés peuvent avoir des boucles et des multi-arcs, voir la figure IV.1.

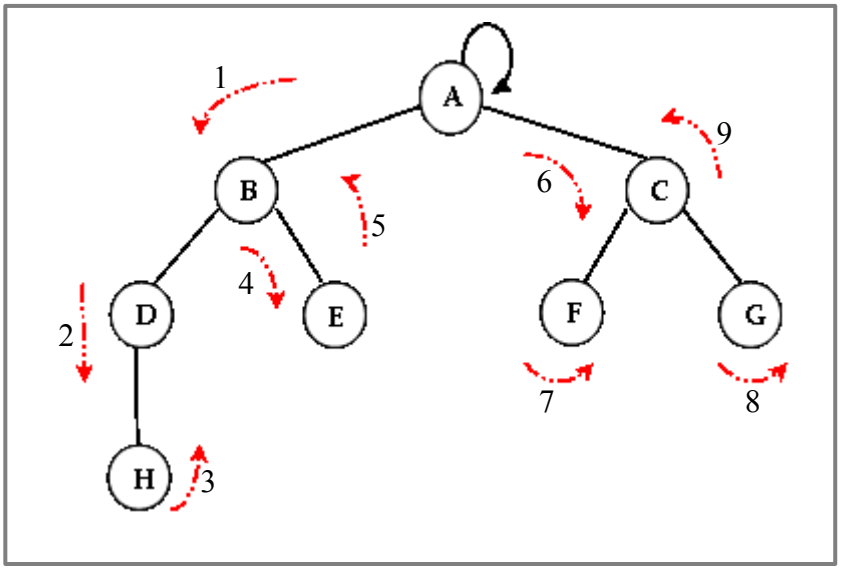
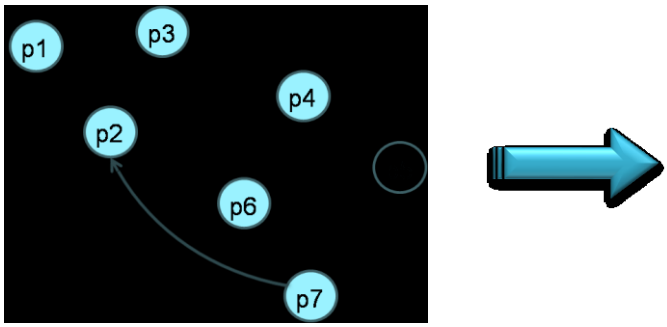


Figure IV. 1 : Parcours en profondeur d'abord.

4.2.1 Sauvegarde de l'toile

Le Web est caractérisé par son ample taille, si on veut faire une petite sauvegarde, ceci va être coûteux en mémoire, pour cela nous avons proposé une solution en java avec les hashtable. Cette dernière permet de représenter notre capture par un arbre, voir Figure IV.2.



G = (P, E)

p1	p2,p3
p2	p6
p2	p4
p4	p5,p6
p5	null
p6	p7
p7	p2

Figure IV. 2 : Représentation d'une capture dans la mémoire avec les hashable.

4.3 Architecture du crawler

Comme tout système, une représentation spécifique pour le fonctionnement et le contrôle est donnée soit sous forme de modèle ou d'architecture. Pour notre Crawler, nous avons proposé une architecture simplifiée qui décrit d'une manière générale le mécanisme de fonctionnement (Voir Figure IV.3).

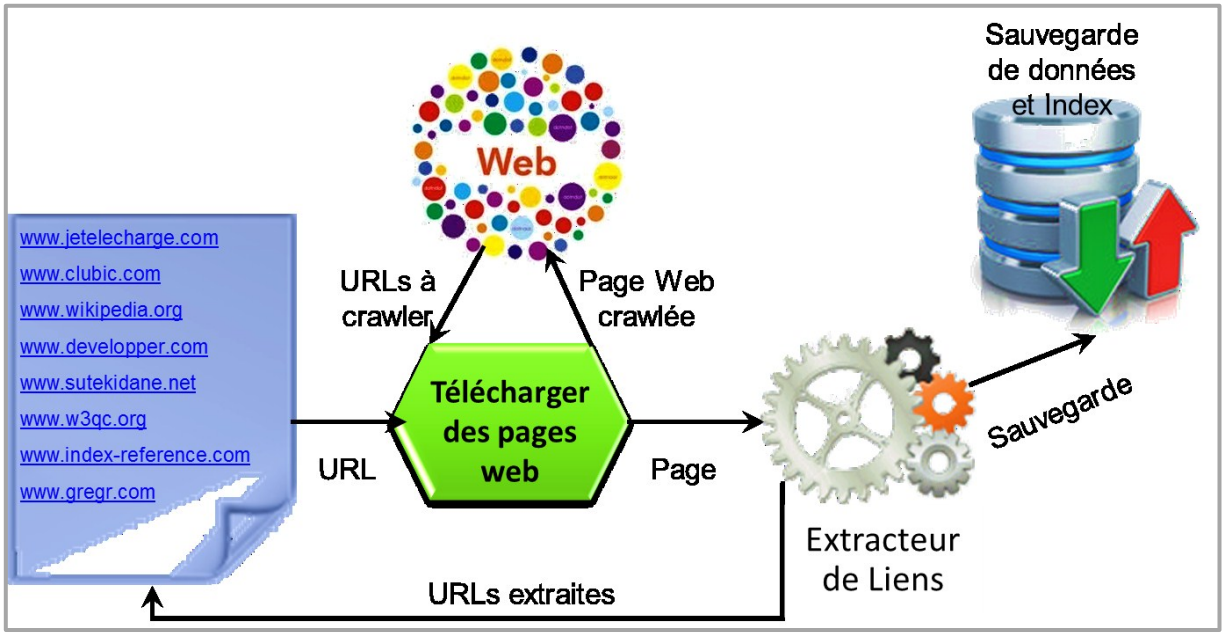


Figure IV. 3 : **Architecture simplifiée de notre Crawler.**

4.4 Établissement des connexions

Dans cette partie, nous détaillons la manière dont laquelle nous avons chargé les pages Web et extrait les liens HyperText contenus ainsi que ses données.

Comme il est décrit dans la section 3, le protocole http permet de transférer des fichiers, essentiellement au format HTML, après avoir les localisés grâce à son URL entre un navigateur (le client) et un serveur Web voir la Figure IV.4.

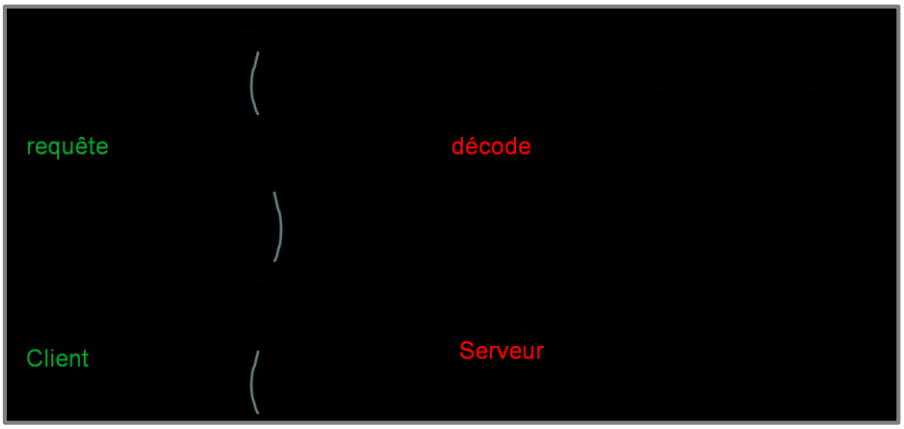


Figure IV. 4 : **Protocol http.**

Après avoir chargé une page web, on parcourt le fichier de notre cache ligne par ligne ensuite, on demande la page par une requête qui ouvre la connexion. Cette requête admet un seul paramètre, une adresse URL.

Si l'adresse URL est valide, alors la page sera téléchargée et sauvegardée temporellement dans un document HTML dans la mémoire. Le résultat est transmis vers un module de sauvegarde de données et d'indexation.

Une page est dite non valide si lors de l'établissement d'une connexion avec elle une exception sera produite sinon le serveur en question retournera la page Web demandée.

Le déroulement de ces étapes est décrit dans le diagramme de séquence illustré par la figure IV.5.

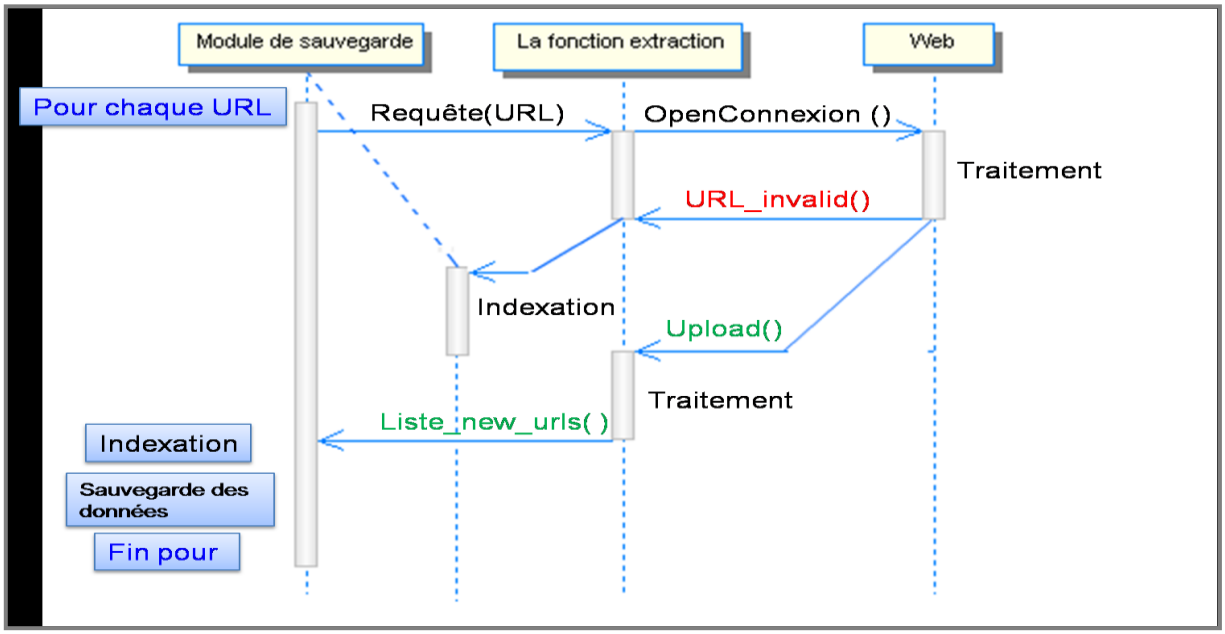


Figure IV. 5 : Diagramme de séquence présentant le crawler.

La sauvegarde des données ne peut pas être faite n'importe comment, puisque nous disposons d'une quantité colossale de données dont on ne peut pas la gérer en état brute, ce qui veut dire que ces données doivent être répertoriées d'une manière organisationnelles. Pour cela, nous avons décortiqué la sauvegarde en plusieurs fichiers distincts, représentés dans la section suivante.

4.5 Fichiers utilisés

Afin d'exploiter et étudier les captures, nous avons utilisé les fichiers suivants :

1- Le fichier ***captur.txt*** : contient toutes les pages Web rencontrées. La Figure IV.6 montre un exemple d'adresses url sauvegardées.

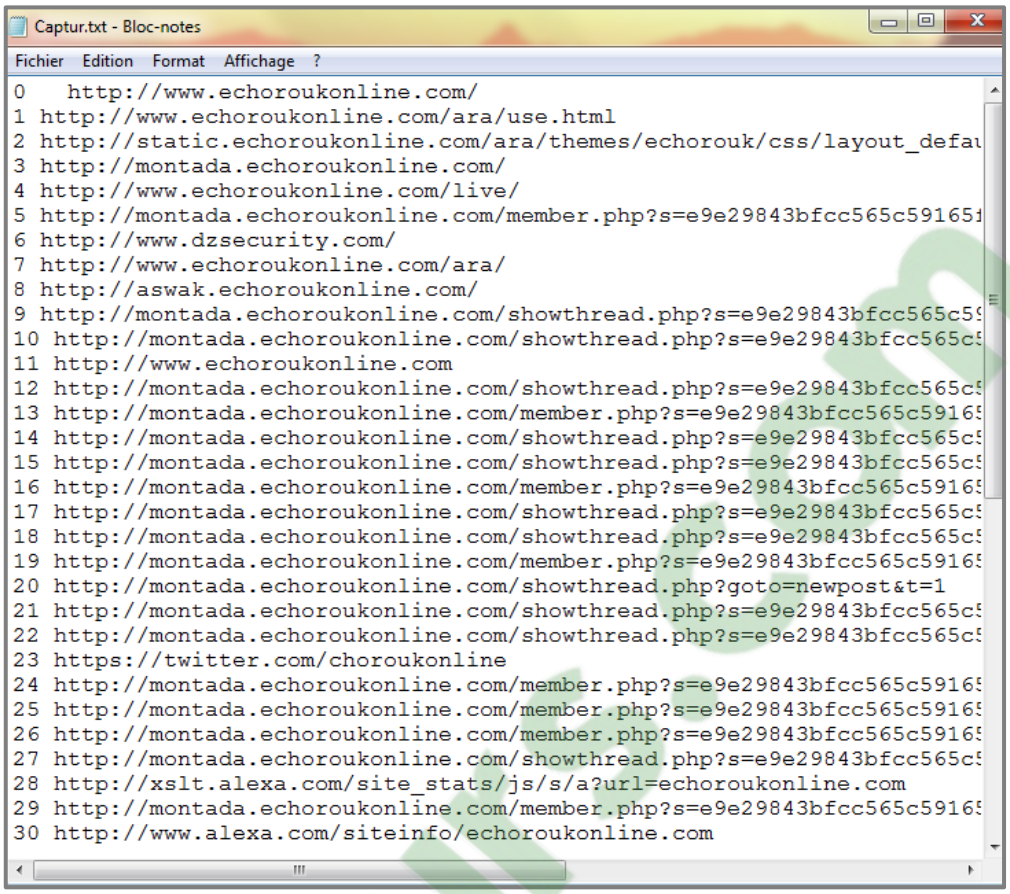


Figure IV. 6 : Exemple de capture.

2- Le fichier ***Index.cwf*** : les adresses url capturées sont filtrées de façon à éliminer les redondances. Ainsi, chaque URL reçoit un identifiant unique ; celui-ci permet de manipuler au mieux les pages Web, de plus un identifiant numérique est plus léger à manipuler. La Figure IV.7 suivante est un exemple d'index.

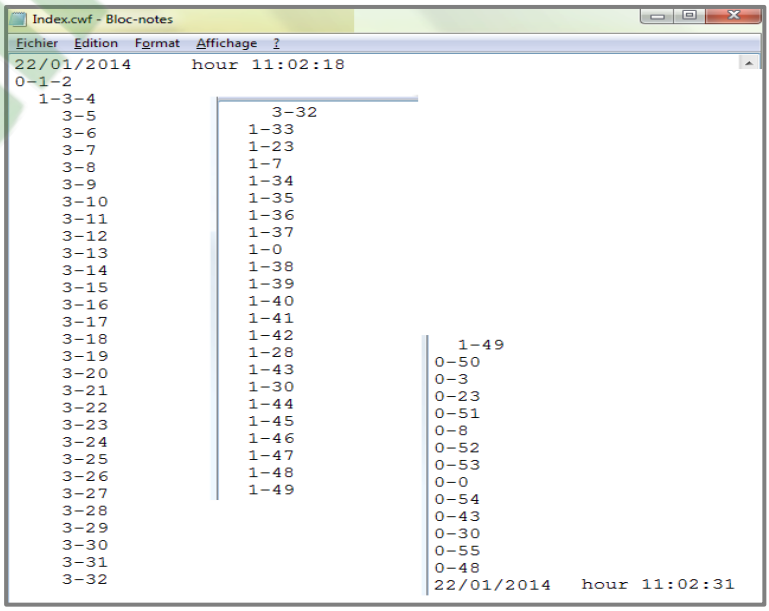


Figure IV. 7 : Une partie de l'index.

La première ligne de ce fichier contient la date début et l'heure début de la capture. Plus encore, la hiérarchie des pages Web est prise en considération et indique que nous faisons une découverte en profondeur d'abord.

3- Le fichier Objet.txt : ce fichier contient tous les nombres de chaque type d'objets découverts dans la page. La Figure IV.8 est un exemple de cette capture.

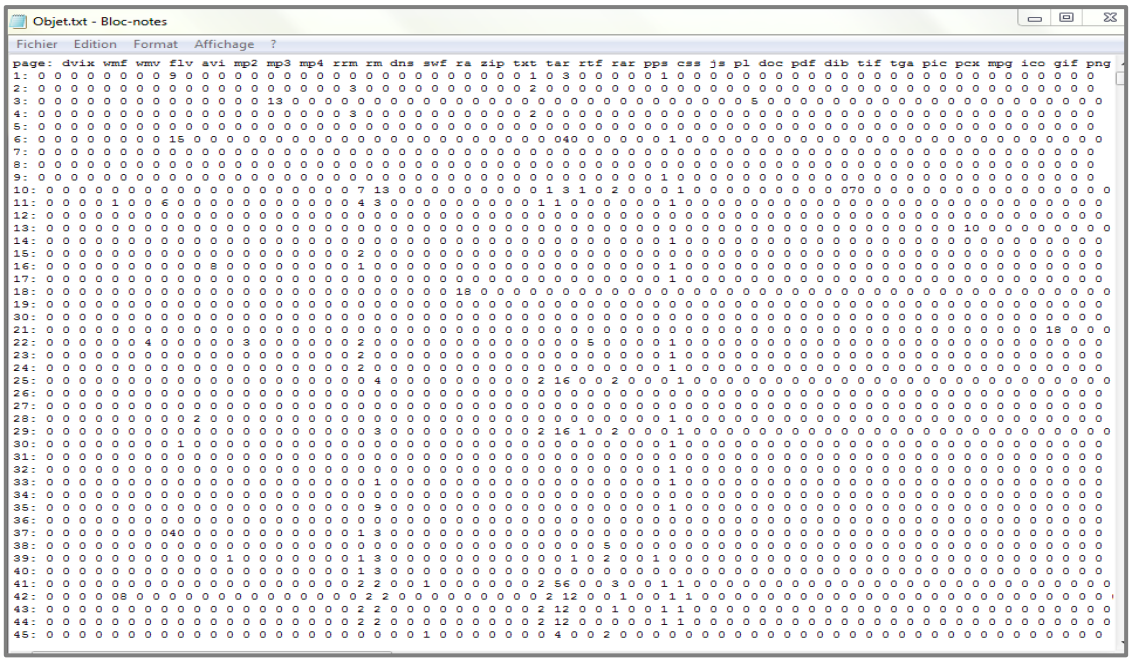


Figure IV. 8 : Une partie du fichier d'objet.

La première colonne de ce fichier est l'identifiant de la page, la première ligne contient tous les types de fichiers rencontrés. Le contenu est le nombre d'apparitions de chaque type dans la page web correspondante.

4- Le fichier Page_texte.txt : le texte extrait à partir des pages Web crawlées est sauvegardé dans ce fichier.

L'organisation des données collectées sont représentées par le lien de la page, suivi du texte extrait dans les lignes qui suivent. Donc à chaque fois qu'on parcoure le fichier et on trouve un lien les lignes au-dessous sont le texte de cette url.

La figure ci-dessous montre une partie d'un texte extrait à partir de la page Web du lien officiel d'echoroukonline.

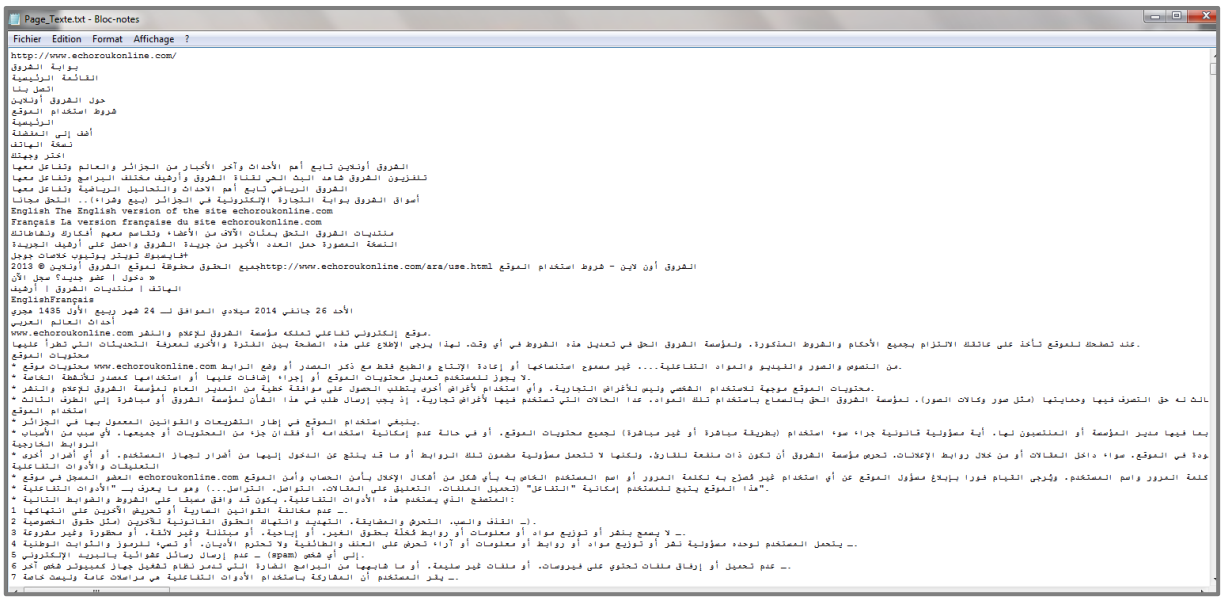


Figure IV. 9 : Exemple du fichier Page-texte.

5- Le fichier Info_Objet.txt : afin de pouvoir faciliter la recherche et l'indexation des objets dans une page Web crawlée et sauvegardée, nous avons créé un fichier qui rassemble tous type d'objets rencontré lors de l'extraction avec son lien source accompagné avec son nom.

À cette heure-ci, malheureusement, nous n'avons pas encore codifié la partie qui concerne l'attribution de noms aux objets, nous avons décidé de le laisser pour les travaux futurs, puisque nous sommes intéressés par le traitement du texte en plus particulier. La figure IV.10 suivante illustre un petit ensemble du contenu de ce fichier où la première ligne indique la colonne des types d'objets, la colonne des liens et le nom des objets.

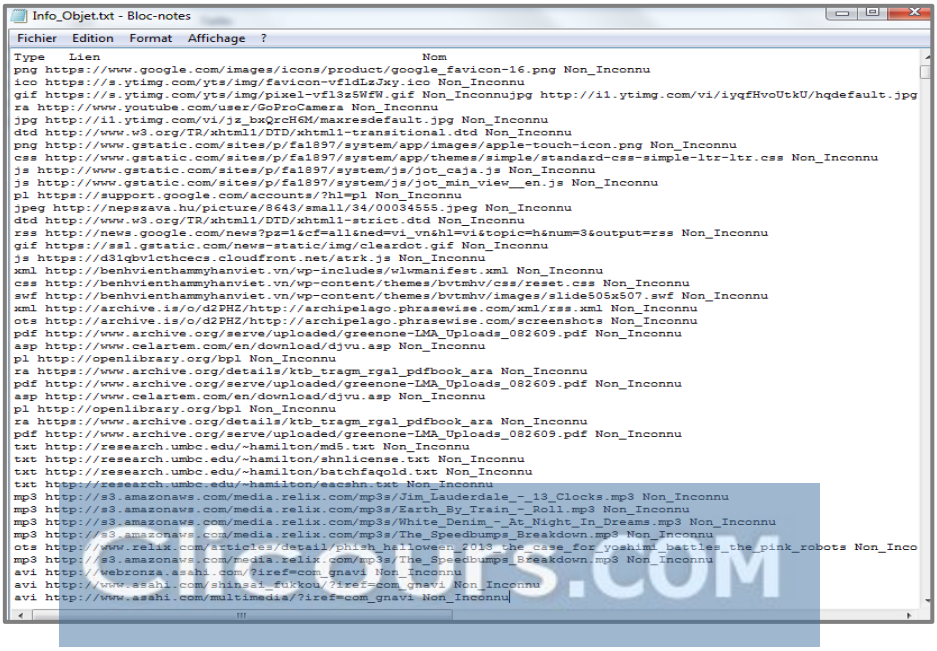


Figure IV. 10 : Une partie du fichier Info_Objets.

5. Problèmes rencontrés et solutionnés

Le domaine des systèmes large échelle demande des machines puissantes en calcul et disposant d'une quantité de mémoire énorme. Ainsi, nous étions contraints de gérer au mieux l'espace mémoire et nous avons utilisé l'espace disque comme mémoire de stockage malgré le coût d'accès élevé. Nous énumérons dans ce qui suit les problèmes rencontrés :

1- Le premier problème est la masse de données collectée lors de l'exploration.

2- Le nombre élevé de liens que peut contenir une page Web. Certaines pages Web contiennent plus de **500** liens. Ceci rend la fonction d'extraction un peu lente.

3- Le crawler peut tomber dans une boucle à cause des pièges dans quelques sites. La résolution nécessite d'intégrer un mécanisme qui libère le crawler.

4- Certaines pages Web contiennent des fichiers de taille énorme et qui prennent beaucoup de temps et d'espace pour leur téléchargement.

5- Plusieurs pages Web ne déclarent pas dans sa structure HTML les attributs « lang », « dir » et « charset ». Comment peut-on chargé une page qui contient un texte écrit en arabe ?

6- Une fois une page rédigée en langue arabe est chargée, elle doit être sauvegardée en mémoire, les éditeurs de texte peuvent le faire mais lors de la réouverture de ces fichiers les caractères en arabe disparaissent et les remplaces d'autre caractères comme ◙, □, ? , §, ¦, ±, ª, Ù, Ø, d'autant plus la direction d'affichage des caractères change de droite à gauche vers de gauche à droite.

Un ensemble de démarches a été adopté afin de remédier ces problèmes qui semblent simples mais ils consomment beaucoup en espace et en temps. Ces problèmes ont été résolus. Dans la suite, nous détaillerons les solutions proposées.

1- Afin d'avoir des flux importants de données chargés à partir du Web, nous avons étendu le tas ou bien le heap de notre machine en configurant la machine virtuelle de Java (Java Virtual Machine : JVM). Explicitement, nous utilisons l'instruction suivante pour cette tâche : Xmx [taille], définit la taille maximum du heap.

2- Après avoir chargé une page Web, nous la traitons, la sauvegardons dans un fichier texte ensuite nous libérons l'espace qu'elle a occupé afin d'en charger une autre et ainsi de suite pour le reste des pages à télécharger. De cette manière nous rendons la fonction d'extraction allégée.

3- Nous sauvegardons les liens déjà rencontrés et si plusieurs fois nous rencontrons le même lien nous sortirons de cette page Web.

4- Un test de taille des pages web téléchargées afin de les comparer avec une valeur e égale à 1 Mo et si la taille de cette page est supérieure à cette valeur, nous récupérons que son adresse url et le sauvegarder.

5- À la place des pages Web qui ne contiennent pas d'attributs : « lang », « dir » et « charset », nous avons déclaré une variable « `charset = Windows-1256`» où une fois nous nous ne rencontrons pas un des trois attributs cités auparavant et spécialement le « charset », nous le remplaçons par ce dernier. Du moment que nous sommes en train de crawler des pages en langue arabe et ce type d'encodage supporte les caractères latins. Dans le même contexte, nous signalons que l'environnement du travail a un grand impact sur le résultat de l'extraction du texte en arabe. Dans les premier temps, nous programma avec eclipse, qui ne supporte point l'encodage arabe où nous avons souffert une longue période en croyant notre démarche est fausse. Ensuite nous avons converti notre espace de travail vers un autre environnement nommé Netbeans où tout était parfait.

6- Une fois que nous avons réglé le problème d'extraction du texte en langue arabe, nous avons eu le souci de le sauvegarder et de le réutiliser pour des études et des traitements ultérieurs d'une manière saine. L'éditeur de texte le plus léger et fiable pour cette mission est bien le *bloc-notes*, mais avant de faire n'importe qu'elle traitement il faut configurer les valeurs binaires des registres en ajoutant une nouvelle clé, portant sur des chaines de caractères multiples en 64 bits. Ensuite accéder au *bloc-notes* pour modifier aussi le mode d'enregistrement des fichiers en UNICODE. La figure suivante montre les changements apportés aux registres et à l'éditeur.

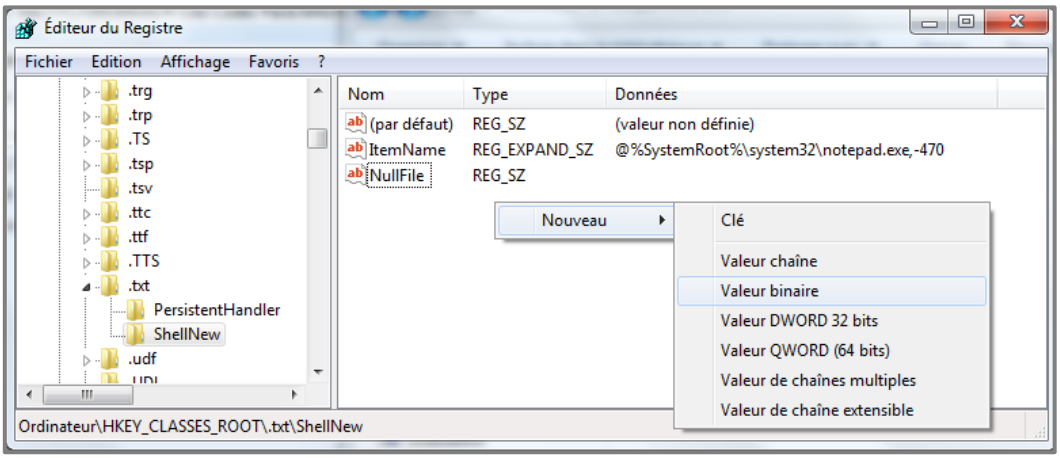


Figure IV. 11 : Ajout d'une nouvelle clé binaire pour le registre .txt.

6. Spécifications de l'application

La figure IV.12 récapitule le fonctionnement de notre application dans un cadre général.

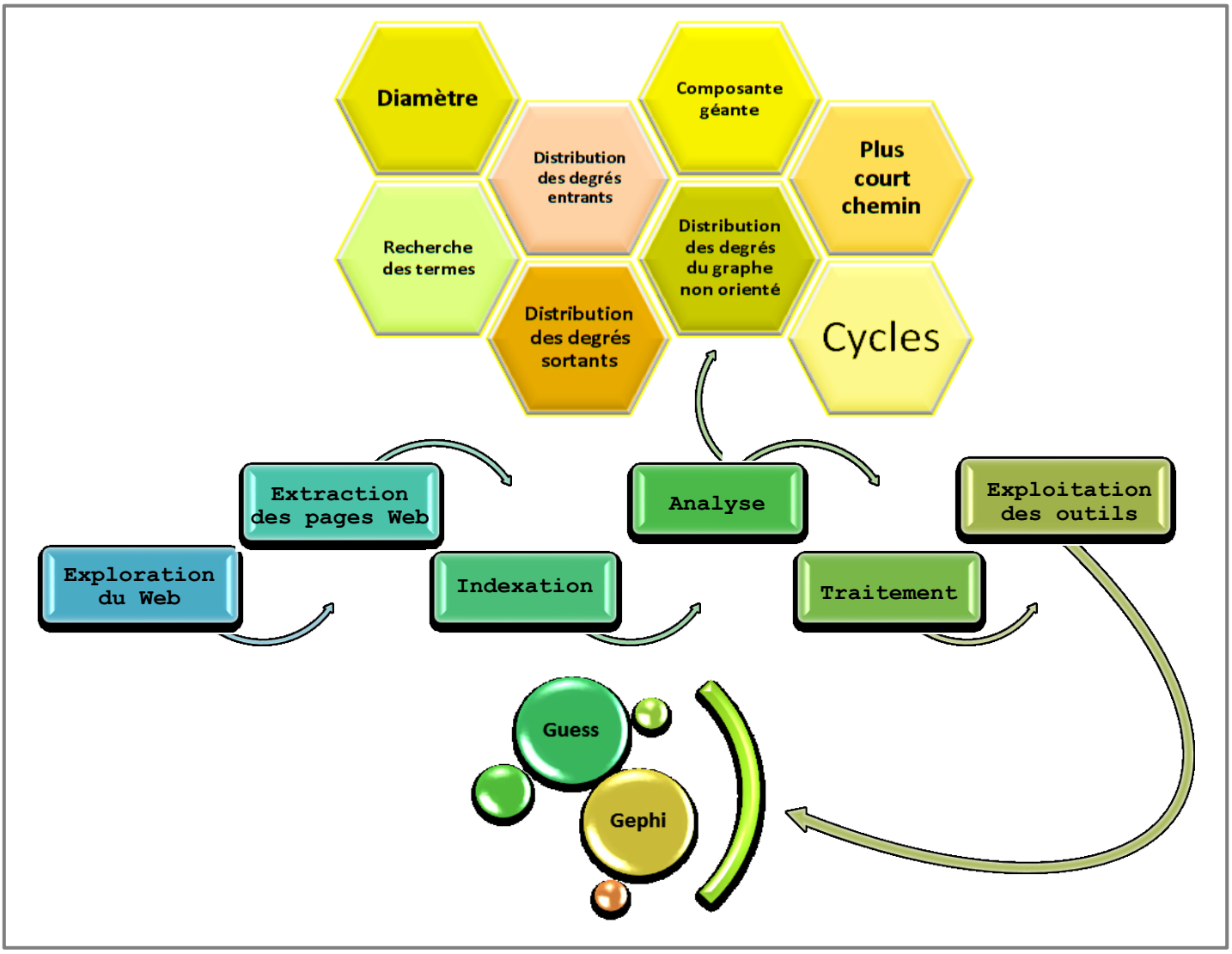


Figure IV. 12 : Croquis général de l'application.

L'application est composée de trois modules : un pour Crawling et deux modules pour la recherche des termes à partir des pages web crawlées et à partir des fichiers en extension pdf.

Afin de donner une flexibilité à notre module du Crawling, nous avons défini plusieurs classes dans le packaging de l'application. La Figure IV.13 représente le diagramme des classes mises au point du module Crawling où :

- La classe Accueil permet d'accéder aux différentes applications.
- La classe Application fait la manipulation de la capture.
- La classe Crawler permet de faire une capture.
- La classe LinkExtract fait l'extraction des liens hypertexte du page web ainsi le texte et les objets.
- La classe Analyse permet l'analyser une capture.
- La classe Statistique fait une représentation graphique des propriétés d'une capture.
- La classe Configuration permet l'organisation ou l'ajout d'une adresse.
- La classe Visualisation fait une représentation graphique d'un graphe à partir d'une capture.

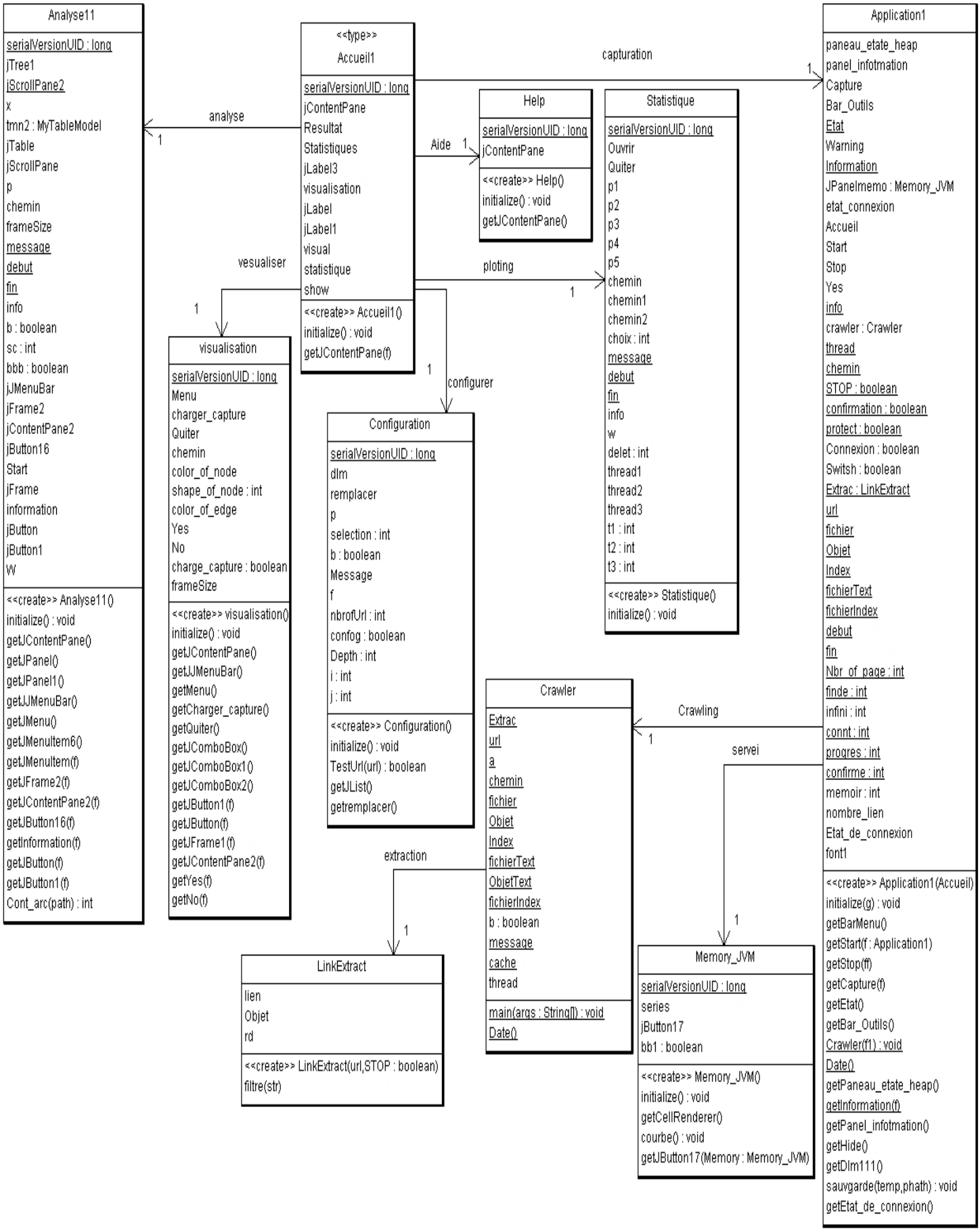


Figure IV. 13 : Digramme de classes du module Crawling.

Pour valider notre travail, nous avons exploité des outils importants dans le domaine, il s'agit de :

1 -Gephi [w19] est un logiciel libre d'analyse et de visualisation de réseaux, développé en Java et basé sur la plateforme NetBeans. Il étudie les captures à partir d'un fichier GDF ou un fichier XML.

2 -Guess [w20] est un outil de visualisation et d'analyse exploratoire des données pour les graphes et réseaux. Il prend en charge l'extension de fichier est *.gdf*. Afin de pouvoir ressortir les clusters ainsi que leurs animations.

7. Traitement des captures

Après avoir terminé la capture, nous avons créé la matrice d'incidence sommet-sommet afin d'étudier les propriétés du graphe. Cette matrice est tirée à partir du fichier Index.cwf. La classe utilisée pour cette tâche est présentée sur le tableau IV.1 alors que Code IV.4 est son code source.

La classe matrice d'incidence

capture	Le ficher en entré représente notre capture.
Matric_page_page (int nbp)	Une fonction qui extrait les fils de chaque page.
La Matrice page_page	La matrice résultante.

Tableau IV. 1 : Création de La classe matrice d'incidence page_page.

Le code Matric_page_page (int nbp):

```
public class matrice_dincidence {
String chemin;
       public  Matrix  Matric_page_page(int nbrp) throws IOException {


              Matrixm=new Matrix(nbrp ,nbrp);
       ArrayList relation= new   ArrayList(0);
       String ligne="";

        BufferedReader fichier = new BufferedReader(new FileReader(chemin));

        while ((ligne = fichier.readLine()) != null) {
         if(ligne.charAt(0)!='d'&&ligne.charAt(0)!='a'){
               int i=0;
         int j=0;
         String Num_page="";

       while(i<ligne.length()){
           if(ligne.charAt(i)!='-')
           {Num_page=Num_page.concat(String.valueOf(ligne.charAt(i)));}
           else{relation.add(Num_page);Num_page="";}
           i++;
           }
```

```
          relation.add(Num_page);


   for(int k=0;k<relation.size()-1;k++)
   {
      String per=  (String) relation.get(k);
      String fils=  (String) relation.get(k+1);
      m.setElement(Integer.parseInt(per.trim()),Integer.parseInt(fils.trim()),1);
      m.setElement(Integer.parseInt(fils.trim()),Integer.parseInt(per.trim()),1);
      }
      relation= new   ArrayList(0);

   }
        }

             return m;
```

Code IV. 4 : **Matrice_dincidence.**

La liste des successeurs est obtenue en fonction de la classe liste_successeur suivante. (tableau IV.2) :

la classe liste successeur

Capture	Le ficher en entrée qui représente notre capture.
getgraphe(String chemine)	Fonction qui extrait pour chaque page leur fils.
Le graphe Hashtable	Arbre en sortie

Tableau IV. 2 : La liste des successeurs.

Le code *liste_successeur*( ):

```
public class Liste_successeurs {
    static Hashtable graphe=new Hashtable();
    static ArrayList fils= new   ArrayList(0);
    static int arc=0;
public Hashtable  getgraphe(String chemin ) throws IOException {
     BufferedReader fichier = new BufferedReader(new FileReader(chemin));
     ArrayList relation= new   ArrayList(0);
     ArrayList feiil= new   ArrayList(0);
     String ligne ;
       while ((ligne = fichier.readLine()) != null) {
              if(ligne.charAt(0)!='d'&&ligne.charAt(0)!='a'){
             int i=0;
                   String p="";
                   int j=0;
            while(i<ligne.length())
            {
                   if(ligne.charAt(i)!='-')
                    {
                          if(ligne.charAt(i)!=' ')
                          p=p.concat(String.valueOf(ligne.charAt(i)));
                    }
                   else{ relation.add(p);p="";  arc++;}
                   i++;
            }
            relation.add(p);
```

```
                String ss1= "" ;
                String ss2="";
                for(int k=0;k<relation.size()-1;k++)
                     {
                    ss1=  (String) relation.get(k);
                       ss2=  (String) relation.get(k+1);
                     if(!graphe.containsKey(ss1))
                       {
                        fils.add(ss2);
                        graphe.put(ss1,fils);
                        fils= new   ArrayList(0);  } else
                        {
                     fils=(ArrayList) graphe.get(ss1);
                     fils.add(ss2);
                     graphe.remove(ss1);
                     graphe.put(ss1, fils);
                     fils= new   ArrayList(0);
                        }
                     }
               feiil.add(relation.get(relation.size()-1));
               relation= new   ArrayList(0);
               }}
          int l=0;

     while(l<feiil.size()){
                if(!graphe.containsKey(feiil.get(l)))
                graphe.put(feiil.get(l).toString(), fils);
          l++;
          }
          return graphe;
          }
     }
```

Code IV. 5: **Code source de liste_successeur.**

A partir de ce stade, nous pouvons commencer l'analyse dans son sens profond. Ceci consiste à faire une étude minutieuse de la capture en développant les points cités dans les objectifs de ce chapitre.

Parmi les algorithmes de la théorie des graphes que nous avons utilisés, on cite l'algorithme de Dijkstra [w21], proposé en 1959. Un algorithme qui permet de déterminer le plus court chemin entre deux sommets d'un graphe connexe pondéré (orienté ou non) dont le poids lié aux arêtes est positif ou nul.

Nous avons adopté et algorithme pour le calcul de la distance moyenne et le diamètre du graphe. L'algorithme IV.1 ci-dessous représente les démarches de l'algorithme de Dijkstra :

```
Soit T la matrice d'adjacence du graphe pondéré.
   void cheminLePlusCourt(void)
      {
        (Soit MinDistance une variable qui contient le poids des arcs)
       (et soit Minimum(x,y) une fonction dont sa valeur est la plus petite entre x
et y.)
       /* Soit v1 de V le nœud d'origine à partir duquel commence le plus court chemin */
        /* Initialiser W et PlusCourteDistance[u] comme suit: */
            W = {v1};
           PlusCourteDistance[v1] = 0;
```

```
        pour (chaque u de V - {v1} ) PlusCourteDistance[u] = T[v1][u];
      /* Successivement élargir W jusqu'à ce que W inclue tous les nœuds de V */
           tantque (W != V) {
      /* trouver le nœud w de V - W à une distance minimale de v1 */
                 MinDistance = infini;
                 Pour (chaque v in V - W) {
                   Si (PlusCourteDistance[v] < MinDistance) {
                       MinDistance = PlusCourteDistance[v];
                                   w = v;
                    }
                 }
               /* ajouter w à W */
                   W = W union {w};
       /* mettre à jour les plus courtes distances aux nœuds de V - W */
               Pour (chaque u de V - W) {
                   PlusCourteDistance[u] =
                     Minimum (PlusCourteDistance[u],PlusCourteDistance[w]+T[w][u]);
              }
           }
```

Algorithme IV. 1: **Dijkstra.**

Nous avons utilisé l'algorithme IV.2 pour la détection de la composante fortement connexe [85]. Il est présenté comme suit :

```
    for i = 1 to n do
                  marquer(i) = 0;
     end for
    Charger_graph();
     for i = 1 to n do
          if marquer(i) = 0 than DFS(i) (ou alors BFS);
           /* restituer le sous graphe visité par DepthFirstSearch (formant une composante
connexe);*/
                end if
     end for.
```

Algorithme IV. 2: **Détection de la composante fortement connexe.**

L'algorithme de parcours en profondeur d'abord de la toile et détection de cycles :

```
PP[G]
                        Pour chaque sommet u ∈ X faire
                            Couleur[u] ← blanc
                            Père[u] ← NIL
                        Temps ← 0
                       Pour chaque sommet u ∈X faire
                               Si Couleur[u] = blanc
                               Alors Visiter_PP[u]
                                  Visiter_PP[u]

Visiter_PP[u]
                                  Couleur[u] ← gris
                                 d[u] ← temps ← temps + 1
                      Pour chaque v ∈ Adj[u] faire
                               Si Couleur[v] = blanc
                               Alors Père [v] ← u
```

```
                        Visiter_PP[v]
                        Couleur[u] ← Noir
                        F(u) ← temps ← temps + 1
```

Algorithme IV. 3: **Parcours en profondeur d'abord de la toile et détection de cycles.**

8. Les études effectuées sur la capture

8.1 Calcul de la distribution des différents degrés existants

Le graphe de la toile se génère petit à petit en fonction des pages Web et leurs indexation entre elles, pour cela nous effectuons une analyse qui permet d'observer comment ces pages sont-elles distribuées. Le code IV.6 présente la manière dont laquelle nous avons calculé la distribution des degrés de ces pages Web.

```
public class degré {

        static Hashtable degre=new Hashtable();
        static ArrayList fils= new   ArrayList(0);

        public static Hashtable getdegre(Hashtable graphe) throws IOException {

              int i=0;
              fils=(ArrayList) graphe.get(String.valueOf(0));

              int    max    =fils.size();
              while(i<graphe.size())
              {
                     if(!degre.containsKey(fils.size()))
                     {
                            degre.put(fils.size(), 1);
                     }
                   else{

         int nbr_fils=Integer.parseInt(degre.get(fils.size()).toString().trim());
                        nbr_fils++;
                     degre.remove(fils.size());
                     degre.put(fils.size(), nbr_fils);

                     }
                     i++;
                     fils=(ArrayList) graphe.get(String.valueOf(i));
               if(i<graphe.size())
               {
                     if(max<fils.size())
                            max=fils.size();
               }
              }
              degre.put("#", max);
       graphe=null;
          System.gc();
              return degre;
       }
   }
```

Code IV. 6: **Code source pour le calcul des degrés sortants.**

8.2 Coefficient de clustering

Pour étudier la connectivité du graphe, nous avons calculé le Coefficient de clustering de chaque page et le Coefficient de clustering global du graphe. Le code source Code IV.7 représente les démarches destinées pour cette tâche.

```
public static Hashtable getCoefficient(String chemin) throws IOException
        {
                list graphe1=new list();
         a=graphe1.getgraphe(chemin);
         probabilite.add(0.0);
        int i=0;
        while(i<a.size())
        {
                fils=(ArrayList) a.get(String.valueOf(i));

                double C=Double.valueOf(fils.size());
                double nbrConnection=0;
                int j=0;
                while(j<fils.size())
                   {int k=0;
                      ArrayListf1= (ArrayList) a.get(fils.get(j));
                        while(k<fils.size())
                        {
                        for(int ii=0;ii<f1.size();ii++)
                          {
if(Integer.parseInt(f1.get(ii).toString().trim())==Integer.parseInt(fils.get(k).toString().trim()
))   { nbrConnection++; }
                             }   k++;
                        }
                        j++;
                           }
                if((C*(C-1))!=0){
                Coefficient.put(i, (nbrConnection)/(C*(C-1)));
                        if(!regroupment.containsKey((nbrConnection)/(C*(C-1)))){
                        coff=coff+(nbrConnection)/(C*(C-1));
                        probabilite.add((nbrConnection)/(C*(C-1)));
                        nbr_of_node++;
                        regroupment.put((nbrConnection)/(C*(C-1)), 1);}
                        else{
                                 Object jjj = regroupment.get((nbrConnection)/(C*(C-1)));
                                 coff=coff+(nbrConnection)/(C*(C-1));
                                nbr_of_node++;
                                int jj=Integer.parseInt(jjj.toString().trim());
                                jj++;
                                regroupment.remove((nbrConnection)/(C*(C-1)));
                                regroupment.put((nbrConnection)/(C*(C-1)), jj);
                                }
                        }
                        else{Coefficient.put(i, 0.0);
                        nbr_of_node++;
                        if(!regroupment.containsKey(0.0)){
                        regroupment.put(0.0, 1);}
                        else{
                                 Object jjj = regroupment.get(0.0);

                        int jj=Integer.parseInt(jjj.toString().trim());
                        jj++;
                        regroupment.remove(0.0);
                        regroupment.put(0.0, jj);
                        }
                }
                nbrConnection=0;
        i++;
                }
                return Coefficient;
        }
    }
```

Clicours.COM

Code IV. 7: Code source pour le calcul du coefficient de clustering des pages.

8.3 Visualisation de la capture

Apprêt avoir effectué l'analyse, nous produisons un fichier GDF afin de le visualiser en fonction du logiciel Guess ou Gephi. À l'aide d'un ensemble d'outils et de palettes, ces derniers permettent d'animer os captures.

Après des recherches et des études intensives, nous avons rencontré les logiciels Gephi et Guess qui permettent l'étude des traces Web capturées. Ainsi, nous l'utilisons pour la confirmation de nos résultats et nous avons trouvé que, effectivement nos analyses sont correctes. Ces logiciels utilisent le fichier GDF illustré dans la figure IV.14.

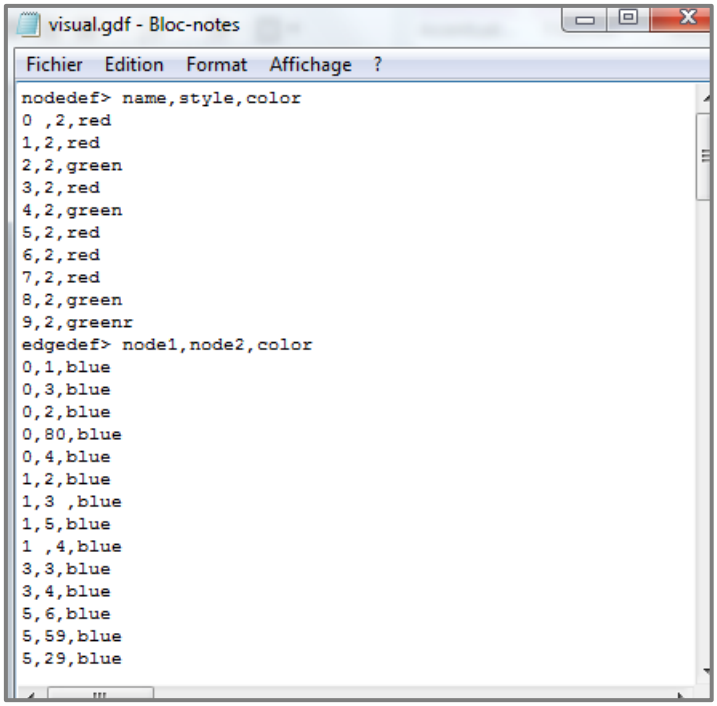


Figure IV. 14: Création de fichier GDF.

Ce logiciel fait aussi l'animation et donne plusieurs spatialisations raffinées du graphe obtenu.

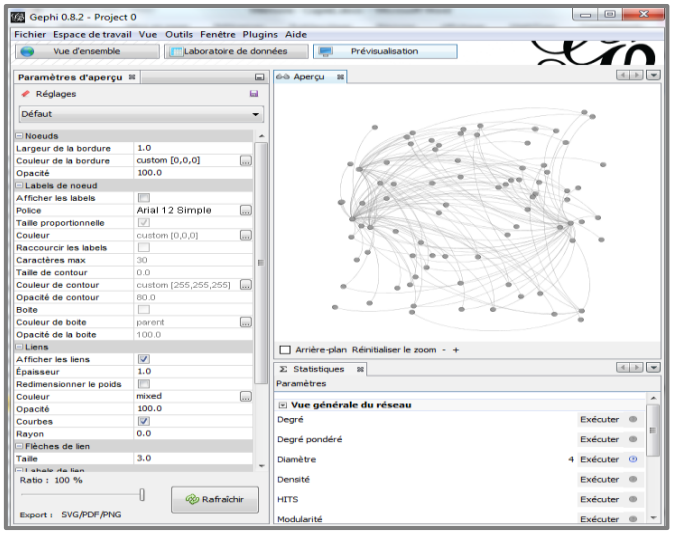


Figure IV. 15 : La plate-forme de l'API Gephi 0.8.2.

9. Conclusion

Ce chapitre regroupe toutes les étapes que nous avons faites pour créer notre crawler web et un moyen d'extraction de texte, spécialement en langue arabe. Nous avons aussi présenté les différents points de vus lors du développement des études des captures. La réalisation elle-même du crawler performant et robuste est un défi fort à relever. Nous estimons que nous avons franchi un grand pas pour devenir autonome en question de capture et d'analyse. Ce chapitre contient aussi un ensemble d'outils qui ont été utilisés pour confirmer que notre analyse est correcte. Ainsi, nous avons abordé certains obstacles que nous avons rencontrés lors de ce parcours, ainsi que la manière dont laquelle nous avons entrepris pour les résoudre.

Le chapitre suivant représente la partie implémentation de notre logiciel.

Chapitre V : Implémentation

1. Introduction

Ce chapitre explique les étapes de notre propre Crawler et les modules d'extraction du texte.

Après avoir lancé le Crawler dans le web, nous avons eu des captures sur lesquelles nous faisons des analyses. Premièrement, afin de vérifier les propriétés, telles que la distance moyenne, le coefficient du clustering et la distribution des degrés. Ainsi le calcul du plus court chemin pour faire ressortir le diamètre de la capture, le plus long chemin afin de calculer la profondeur de la capture, et bien sûr les composantes fortement connexes et la détection des cycles. Nous avons étudié aussi la distribution des objets contenus dans le web car son organisation nous a intrigués. Deuxièmement, collecter le maximum de pages web pour extraire les données afin de les indexer, dans ce travail, nous nous sommes concentrés sur l'extraction du texte pour implémenter dans le futur proche un système de Recherche d'informations robuste et effectif.

2. Environnement

L'environnement de notre travail est *Microsoft Windows 7 avec* EDI Java (Environnement Intégré de Développement) eclipse *IDE (Version: Indigo Service Release 2), eclipse (version Kepler Service Release 1) et Netbeans IDE* (*version7.4 Patch* 3) qui sont très souples et performants, en plus ils sont distribués gratuitement par *Sun Microsystems (*Open Source*)*.

2.1 Choix du langage de programmation

Il est indispensable de donner dans ce chapitre nos principales motivations quant au choix de Java comme langage de programmation, plutôt que d'autres comme le C++ par exemple.

Java ne cesse de prouver sa force de part, son importante utilisation par de plus en plus des applications web) à travers le monde.

Java est un langage de programmation orienté objet développé par SUN et dont le kit de développement est téléchargeable gratuitement. Comme principaux avantages de Java nous pouvons citer :

Java est un langage « objet », par opposition à C++ qui lui, est « orienté objet » et qui autorise la programmation procédurale. Tous les éléments de Java, à l'exception de quelques types de base tels que les nombres, sont des objets. La conception orientée objet présente de nombreux avantages pour les projets sophistiqués, elle a remplacé les techniques structurées précédentes.

Il est beaucoup plus facile d'obtenir du code sans erreur à l'aide de Java qu'avec C++. Selon certaines estimations, Le code C++ contient au moins une erreur toutes les cinquante lignes. Les concepteurs de Java ont beaucoup réfléchi à la raison pour laquelle le code C++ contient autant d'erreurs. Cette réflexion les a amenés à ajouter dans java des fonctions destinées à éliminer la possibilité de créer du code contenant les types d'erreurs les plus courantes.

Le choix des auteurs de Java a été d'abandonner des éléments mal compris ou mal exploités des autres langages tels que la notion de pointeurs (pour éviter les incidents en manipulant directement la mémoire), l'héritage multiple et la surcharge des opérateurs. Par ailleurs, toutes les variables sont typées et il n'existe pas de conversion automatique qui risquera une perte de données. Si une telle conversion doit être réalisée, le développeur doit obligatoirement utiliser un *cast* ou une méthode statique fournie en standard pour la réaliser.

L'allocation de la mémoire pour un objet est automatique à sa création et Java récupère automatiquement la mémoire inutilisée grâce au *Garbage Collector* qui restitue les zones de mémoire laissées libres à la destruction des objets.

La sécurité fait partie intégrante du système d'exécution et du compilateur. Un programme Java qui plante ne menace pas le système d'exploitation. Il ne peut pas y avoir d'accès direct à la mémoire.

En outre, Java présente l'intérêt de pouvoir être utilisé comme un langage de programmation polyvalent pour développer des logiciels capables de fonctionner sur différentes plates-formes.

2.2 La machine utilisée

Dans le développement de cette application nous avons utilisé un micro-ordinateur de type *«PC portable HP Pavilion dv7 Notebook*» avec **6.00 Go** de mémoire vive, un système d'exploitation **64 bits**, un processeur « *Intel(R) Core*™ i7-3610QM @ 2.30 *GHz »* et une carte graphique NVIDIA modèle **630M** capacité **2 Go**.

2.3 Les Bibliothèques

2.3.1 Flanagan

C'est une bibliothèque contenant un ensemble de règles, d'algorithmes de statistiques et de mathématiques, comme la loi de puissance, la loi binomiale,…etc.

2.3.2 Jfreechart

FreeChart est une bibliothèque open source qui permet de d'afficher des données statistiques sous la forme de graphiques. Elle possède plusieurs formats dont le camembert, les barres ou les lignes et propose de nombreuses options de configuration pour personnaliser les graphiques. Elle peut être utilisée dans des applications standalone ou des applications web et permet également d'exporter le graphique sous la forme d'une image.

3. Fenêtrages

Dans la suite de ce chapitre, nous aborderons les différentes fenêtres de notre application ainsi que les afférentes statistiques. Nous commençons par la description de la fenêtre principale, les fenêtres conçues pour le crawling ensuite, nous passons aux fenêtres créées pour l'extraction du texte.

3.1 La Fenêtre principale

L'idée est de fournir un système de fenêtrage qui reflète une certaine flexibilité au logiciel, ceci rend les accès aux déférentes applications fluides.

L'application en générale se divise en deux parties, la partie gauche qui contient un menu de déroulement avec huit applications :

La première est celle du crawling. La deuxième c'est la fenêtre de configuration d'une capture. Le help est donné dans la troisième. Pour la quatrième fenêtre représente l'analyse d'une capture. La cinquième est développée pour les statistiques. La sixième est utilisée pour des tests on line, la septième fenêtre c'est pour afficher les résultats des captures déjà analysées, la dernière permet de visualiser une capture afin qu'une interprétation soit donnée.

La deuxième partie de l'application est dédié pour l'extraction du texte soit à partir des pages crawlées ou des documents sous format pdf.

Figure V. 1: Fenêtre principale.

3.2 La Fenêtre crawler

Lorsqu'on lance le crawler avec le bouton Start Capture, les informations se défilent dans un panneau textuel qui indique l'adresse URL de chaque page en cours d'extraction et un autre panneau indique : le nombre de pages capturées, le nombre total d'URLs et la vitesse du

crawling. Un panneau « Heap Memory Usage » affiche l'état de la mémoire, si le heap et inférieur à un seuil, une icône rouge s'affiche pour dire qu'il y a une insuffisance de mémoire.

Pour arrêter la capture, un simple clic sur le bouton Stop et une confirmation par oui de la boite de dialogue qui s'affiche juste après le clic, la capture s'arrête, on récolte les pages sauvegardées dans le cache temporairement. Sinon on continue notre travail.

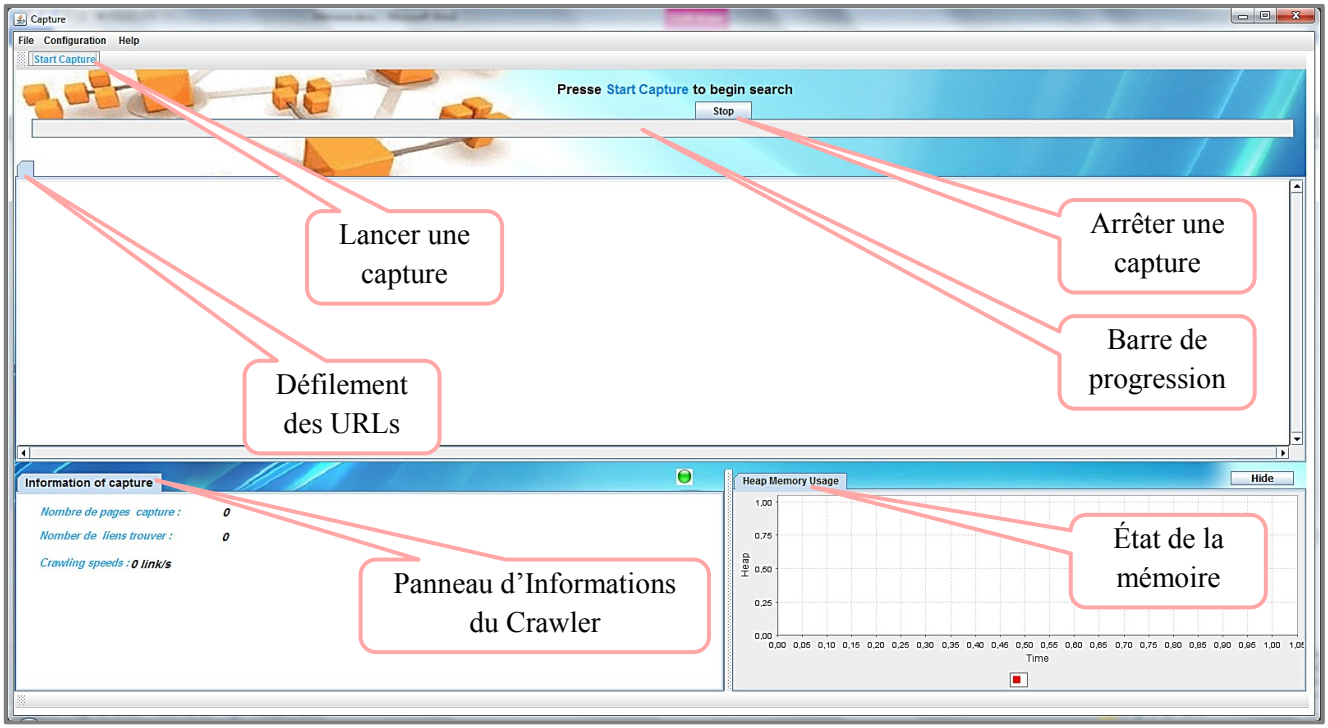


Figure V. 2: **Lancement du crawling.**

3.3 La Fenêtre d'Analyse d'une capture

Cette fenêtre permet d'analyser les captures en mode hors ligne. Le chargement d'une de ces captures se fait à partir de la barre de menu, et l'analyse s'effectue par le bouton Start. Si l'analyse est terminée, une boite de dialogue s'affiche et donne la main à l'utilisateur pour confirmer une sauvegarde des résultats. Les informations de dates de capture sont aussi affichées.

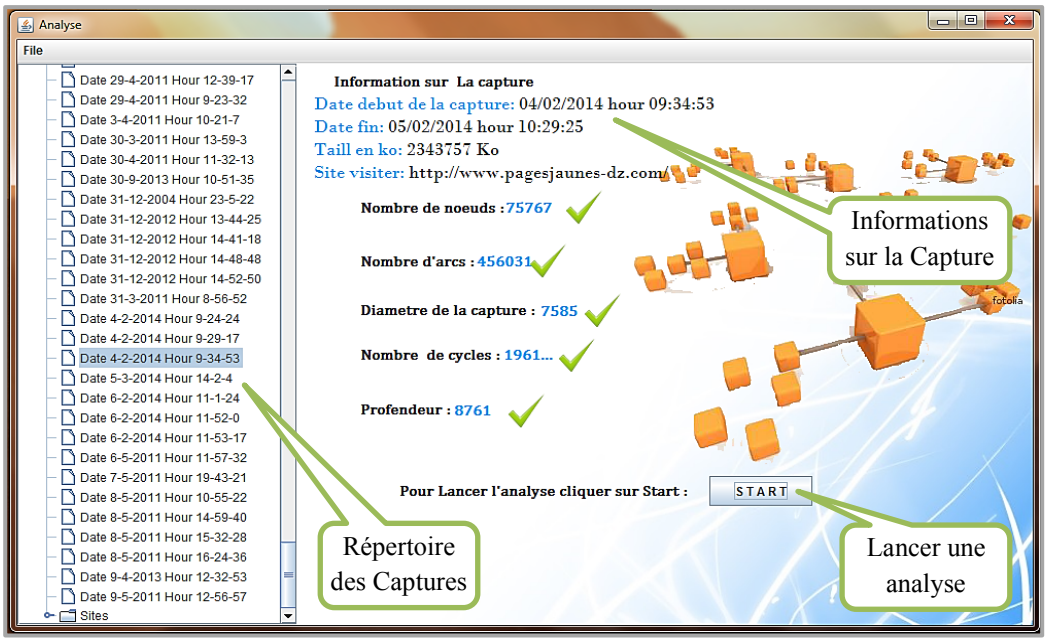


Figure V. 3: Lancement d'une analyse.

3.4 La Fenêtre de Configuration

Elle permet la configuration du cache initial. Lors de l'ajout d'une adresse URL une vérification de la validité de cette adresse est faite. Le but est d'éviter la pollution du cache par des adresses erronées.

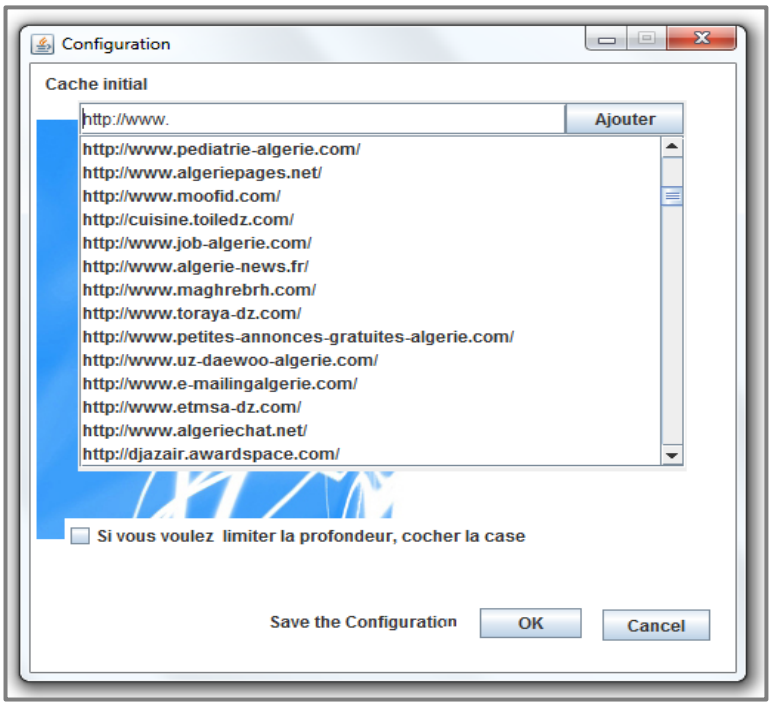


Figure V. 4 : Configuration du cache.

3.5 La Fenêtre Statistiques

Elle permet d'afficher les différentes fenêtres qui montrent les résultats obtenus :

- ✓ Distribution des degrés sortants, entrants et les degrés du graphe non orienté, ainsi que la valeurλ, l'exposant de la loi de puissance.
- ✓ Il est clair sur cette figure que la plus part des pages web possèdent un degré sortant faible (moins de dix fils) alors que peu de pages possèdent un degré fort (**130** fils). Ceci joint la théorie donnée concernant la distribution en loi de puissance [86] où la courbe est à queue lourde.

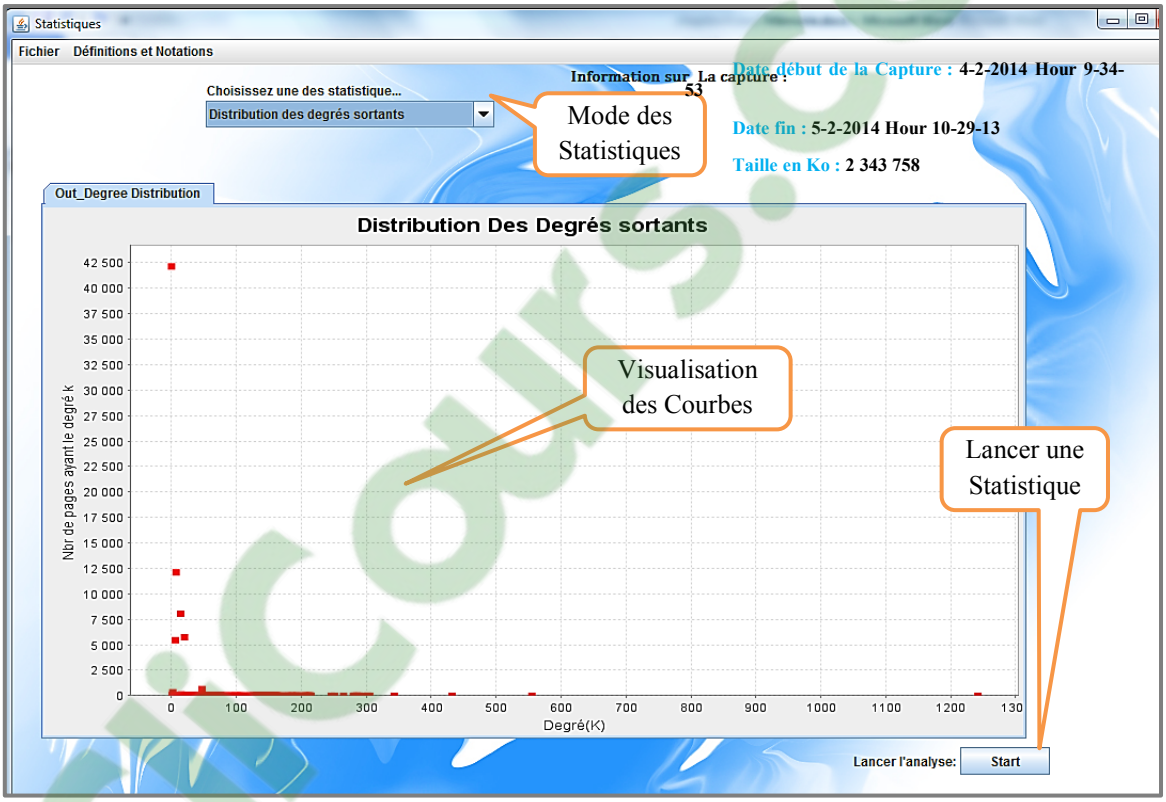


Figure V. 5: **Distribution des degrés sortants on line**.

3.6 La Fenêtre show

Permet de charger une capture afin de chercher les successeurs d'un site ainsi d'identifier un index donné, dont l'utilisateur fait rentrer un numéro et la fenêtre détecte son lien.

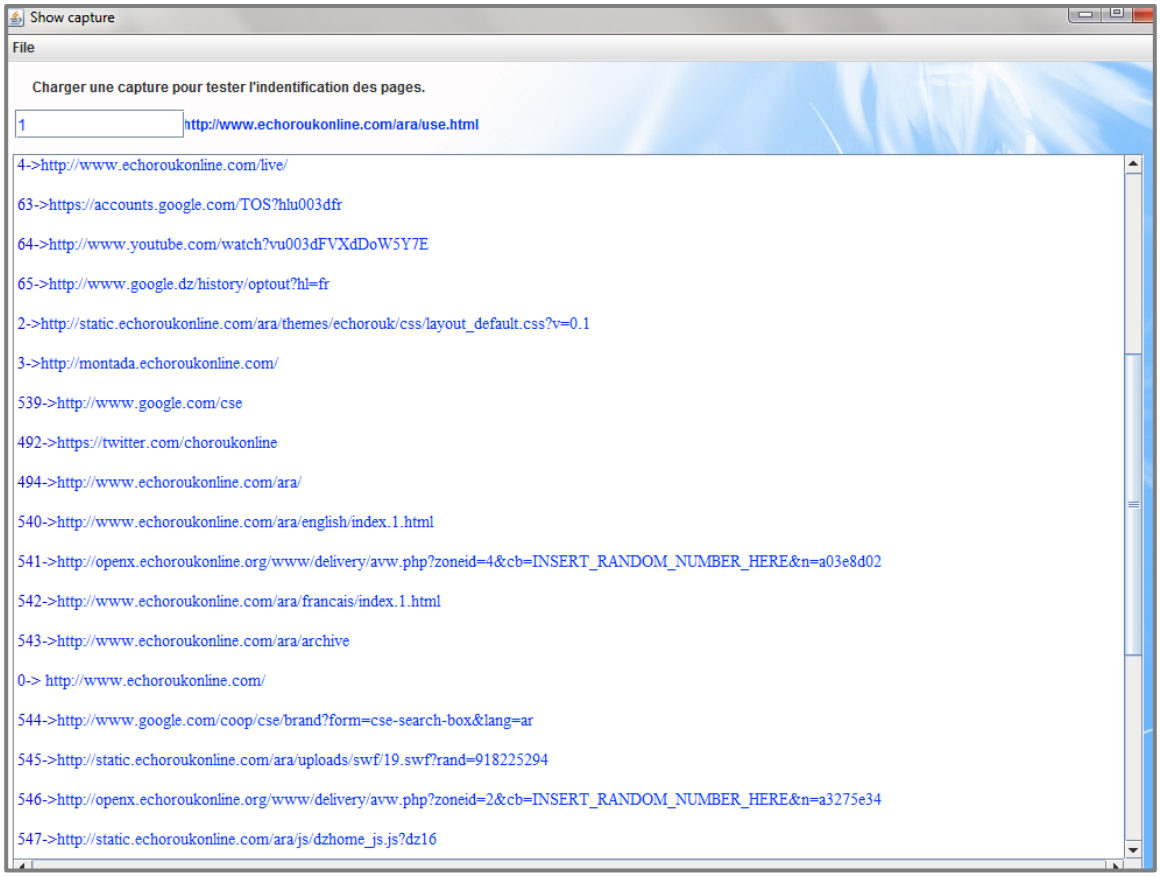


Figure V. 6 : Fenêtre du test on-line.

3.7 La Fenêtre HTML report

Cette fenêtre permet d'afficher tous les résultats des captures analysées.

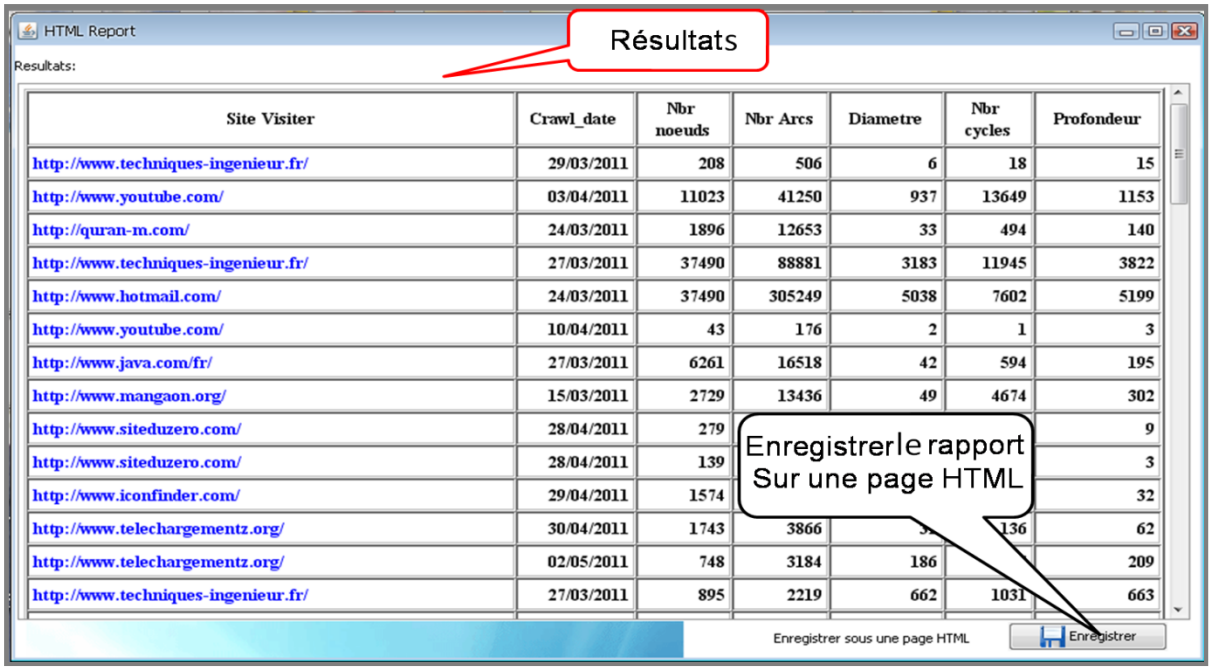


Figure V. 7: Fenêtre résultats.

3.8 La Fenêtre Visualisation

Cette fenêtre permet de modéliser les captures par une représentation graphique afin de donner une allure approximative du graphe obtenu elle donne le choix entre les outils graphiques : Guess ou Gephi.

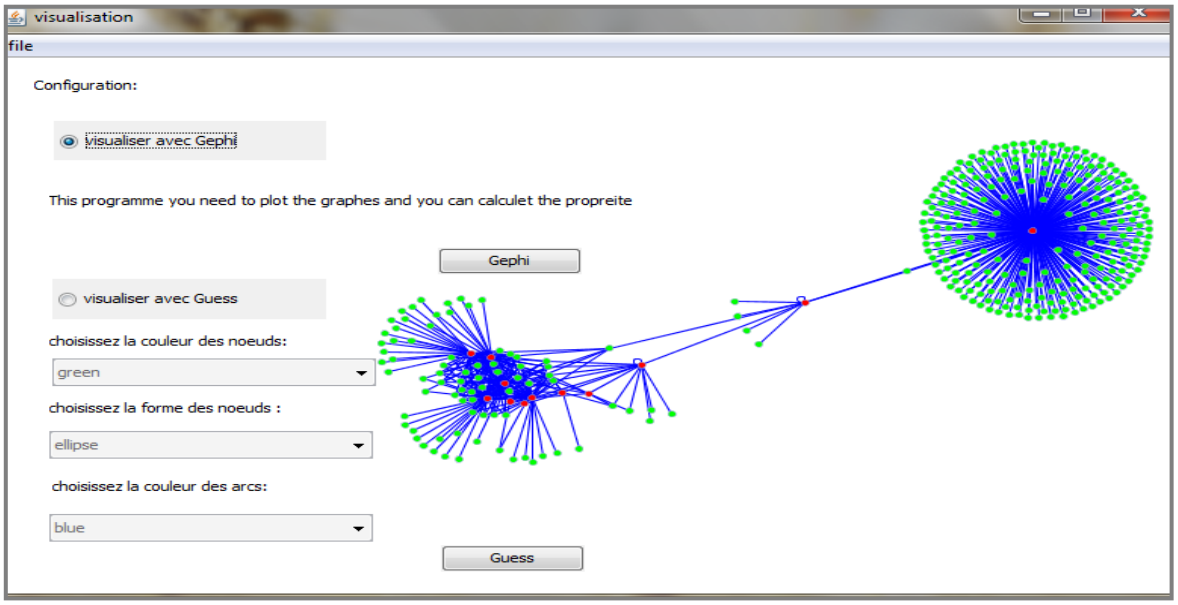


Figure V. 8: Visualisation.

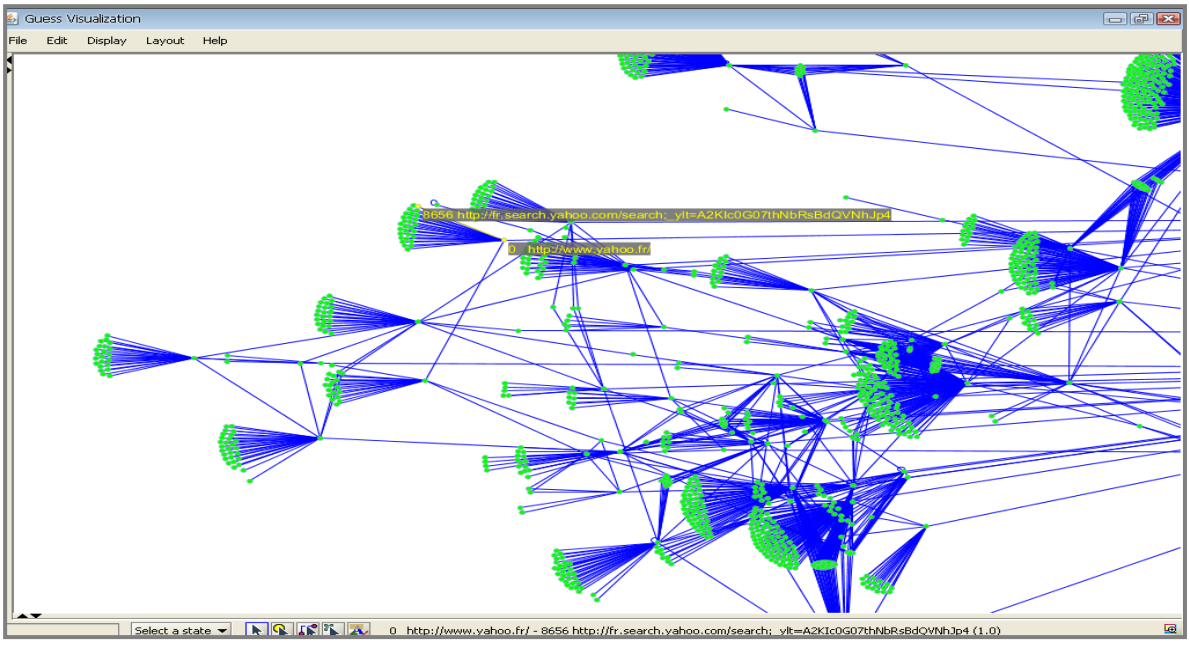


Figure V. 9: Guess visualisation.

Clicours.COM

Dans la suite, nous révélerons les fenêtres de l'extraction du texte.

3.9 La Fenêtre Extraction Web

Cette fenêtre est un index en elle-même, ici nous pouvons lister tous les mots rencontrés dans une page Web.

Le bouton *ouvrir* permet de charger une capture, le bouton *traiter* s'infiltre dans le dossier de la capture pour accéder au fichier contenant le texte et il commence à remplir le tableau par la date du crawl ; il liste les liens capturés.

Pour chaque lien rencontré, il extrait les termes avec leur nombre d'apparitions dans le but de créer un index pour notre futur système de Recherche d'Information.

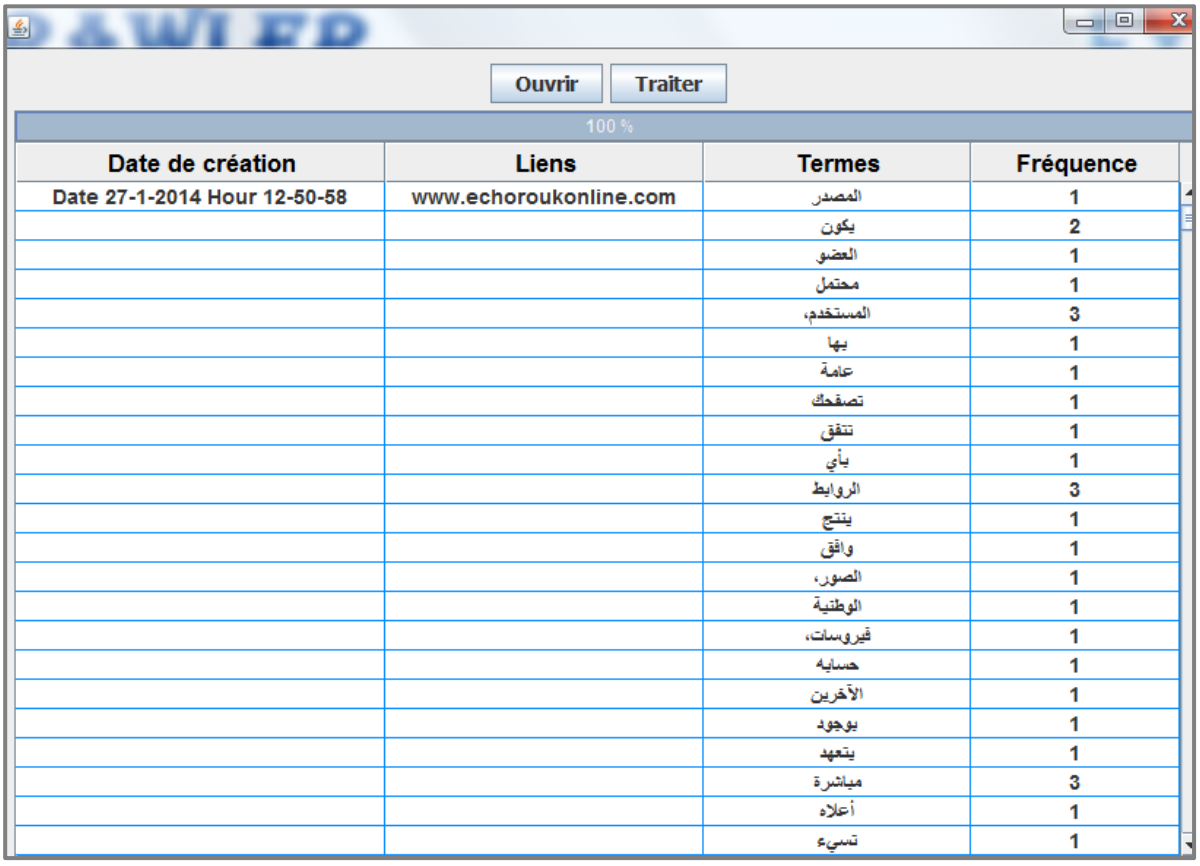


Figure V. 10 : Fenêtre Extraction Web.

3.10 La Fenêtre Extraction PDF

L'extraction et l'indexation du texte est représentée dans la figure V.11, cette fenêtre permet aux utilisateurs de choisir un dossier contenant des documents au format pdf avec le bouton *choisir* et saisir un terme dans la zone *pattern* ainsi lancer la recherche.

Les documents contenant ce terme s'affichent avec les numéros de pages où on peut trouver ce terme recherché. Dans la zone droite, une visualisation du document pdf.

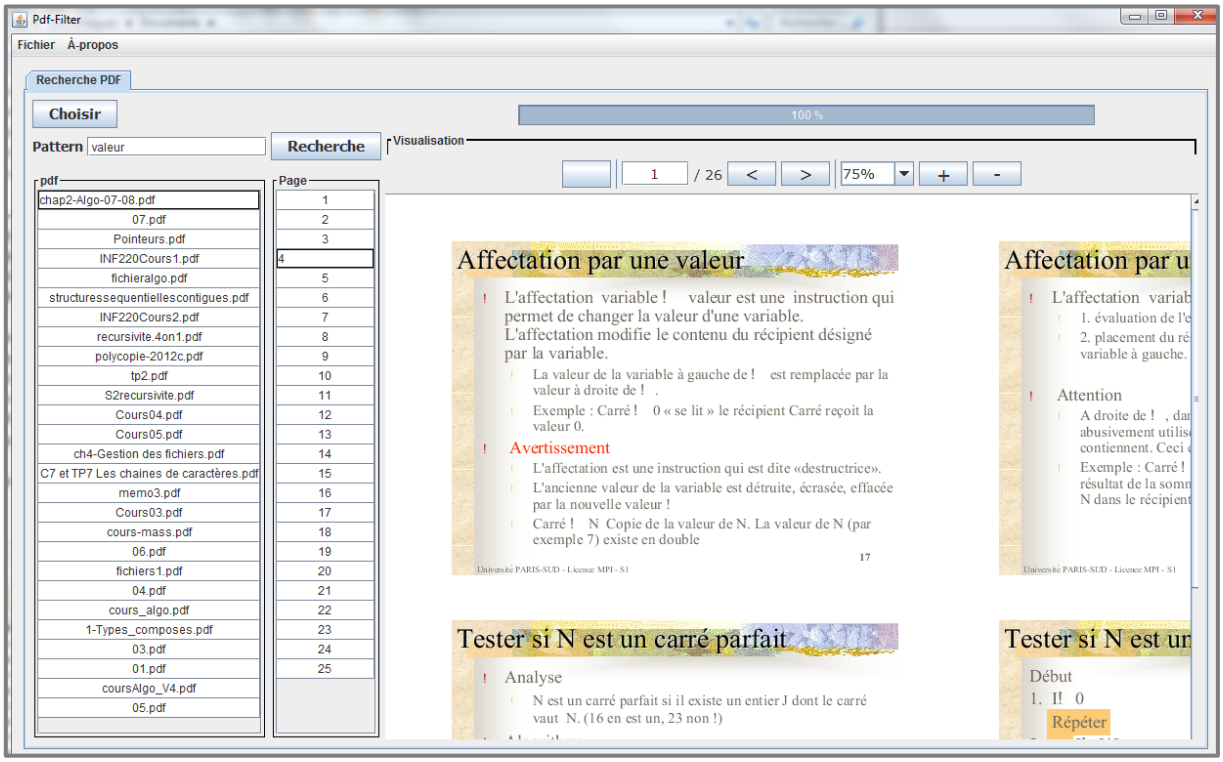


Figure V. 11 : La fenêtre Extraction PDF

Dans la partie suivante, nous exposerons les différents résultats obtenus.

4. Résultats du crawling

Nous avions effectué plusieurs captures qui sont représentées dans un tableau « Capture Analysée ». Ce tableau récapitule le nombre des pages, le nombre des arcs, le diamètre, le nombre de cycles ainsi la profondeur qu'un crawl a touchée lors d'une seule capture.

HTML Report

Site Visiter	Crawl_date	Nbr noeuds	Nbr Arcs	Diametre	Nbr cycles	Profondeur
http://www.techniques-ingenieur.fr/	29/03/2011	208	506	6	18	15
http://www.youtube.com/	03/04/2011	11023	41250	937	13649	1153
http://quran-m.com/	24/03/2011	1896	12653	33	494	140
http://www.techniques-ingenieur.fr/	27/03/2011	37490	88881	3183	11945	3822
http://www.hotmail.com/	24/03/2011	37490	305249	5038	7602	5199
http://www.youtube.com/	10/04/2011	43	176	2	1	3
http://www.java.com/fr/	27/03/2011	6261	16518	42	594	195
http://www.mangaon.org/	15/03/2011	2729	13436	49	4674	302
http://www.siteduzero.com/	28/04/2011	279	553	4	15	9
http://www.siteduzero.com/	28/04/2011	139	142	2	0	3
http://www.iconfinder.com/	29/04/2011	1574	1945	15	85	32
http://www.telechargementz.org/	30/04/2011	1743	3866	32	136	62
http://www.telechargementz.org/	02/05/2011	748	3184	186	227	209
http://www.techniques-ingenieur.fr/	27/03/2011	805	2210	662	1031	663

Figure V. 12: **Analyse.**

4.1 La distance moyenne

D'après notre calcul de la distance moyenne et le nombre de cliques existantes, nous avons généralement trouvé que le nombre de cliques est égal à **19** cliques. Ce calcul est gourmand en temps et en mémoire. Exemple le site visité (http://www.techniques-ingenieur.fr/).

Distance moyenne = 18.34 nœuds≈19 pages web.

4.2 Étude des distributions des degrés du graphe non orienté

Après avoir fait plusieurs statistiques sur la distribution des degrés, nous avons trouvé que cette distribution suit une loi de puissance. Sur le graphe de la Figure V.13, on lit que le degré 8000 est possédé par deux pages Web. La majorité des pages sont de faibles degrés ce qui explique la forme queue lourde du graphique. Le calcul du paramètre λ a révélé qu'il vaut $2.106 \in [2,3]$; ce qui joint [10]

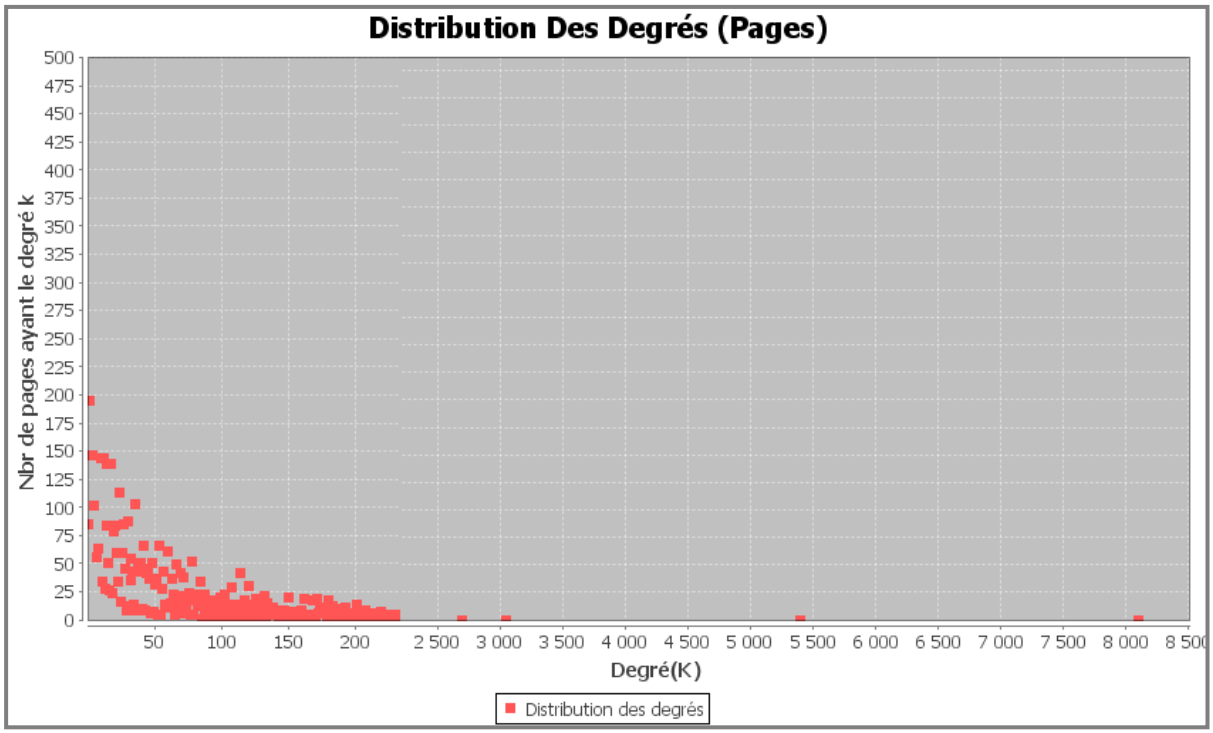


Figure V. 13: Distribution en loi de puissance du graphe non orienté.

Les figurent V.14 et V.15 montrent effectivement que notre démarche est correcte et effectivement nous joignions la théorie des graphes d'interaction car les courbes obtenues sont à queue lourde et que les paramètres des lois de puissances sont toutes dans l'intervalle [2,3].

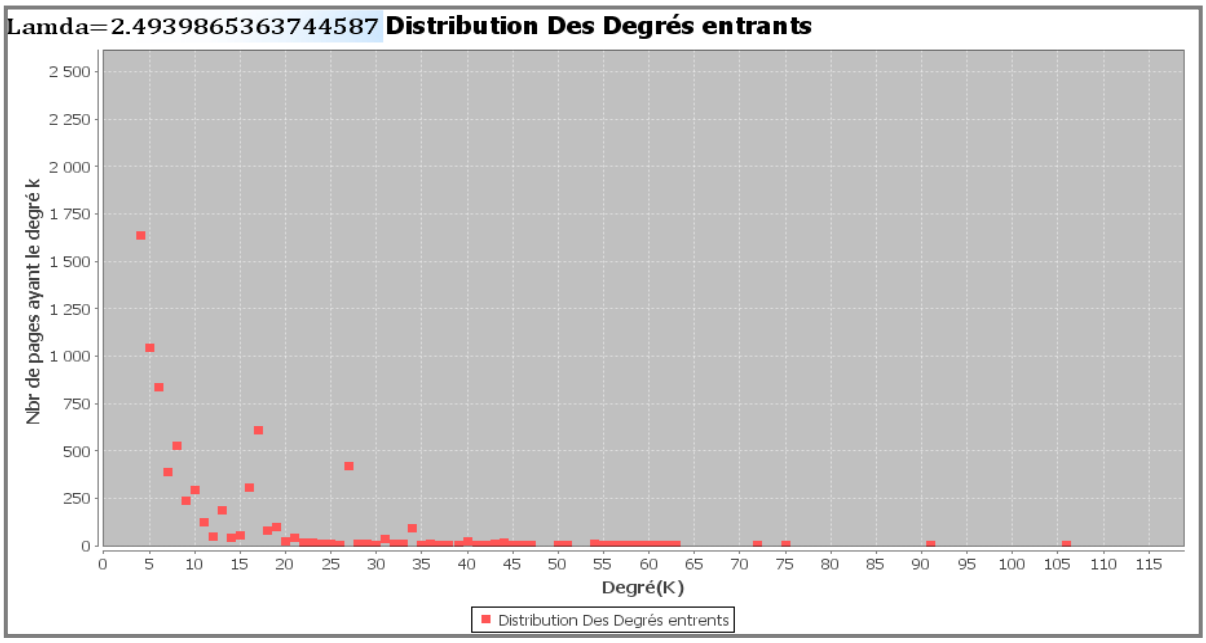


Figure V. 14: Distribution des degrés entrants.

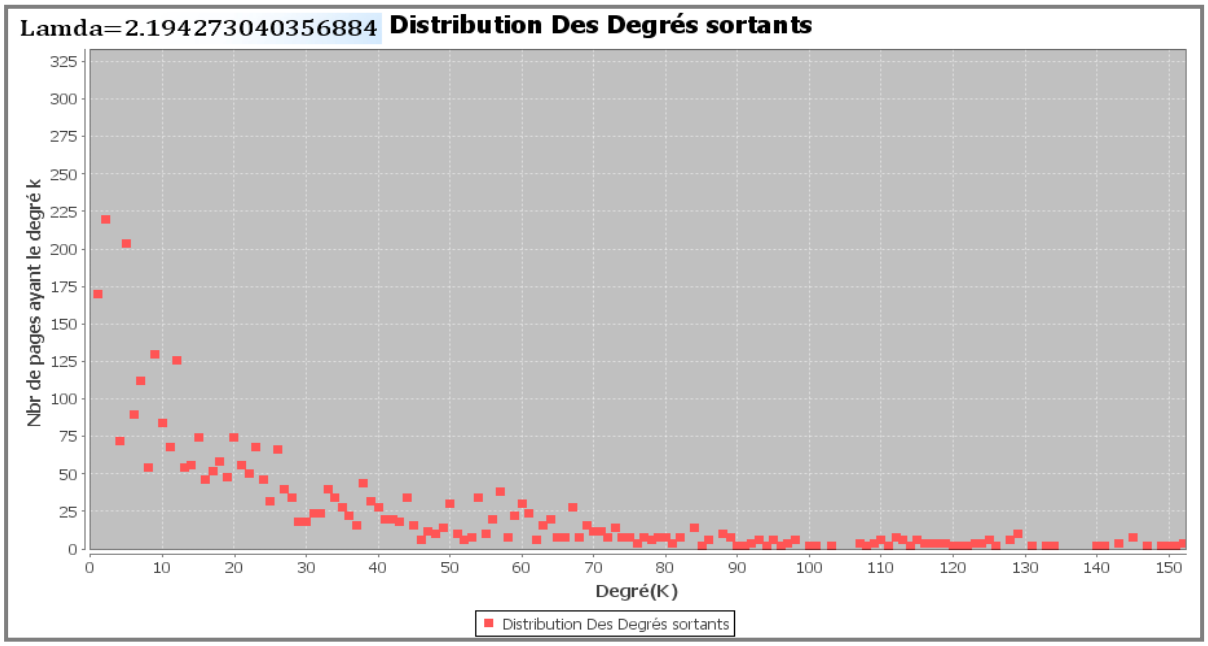


Figure V. 15: Distribution des degrés sortants.

4.3 Coefficient de clustering

La théorie des graphes d'interaction montre effectivement que vu la largeur de Web, le coefficient du clustering reste plus ou moins faibles dans l'ensemble mais localement un ensemble important possède un degré plus ou moins fort. Alors que la théorie de small world montre que le coefficient global est très fort quoique les graphes small world sont plutôt théorique mais point de vue local, il peut être vérifié.

Figure V. 16: **coefficient de clustering.**

4.4 Résultats de la collecte des objets

La Figure V.17 montre la répartition des types d'objets rencontrés dans cette analyse. Nous remarquons que les objets de type **dtd** est le plus rencontré car ce type de fichiers est très utilisés pour la conception des sites web.

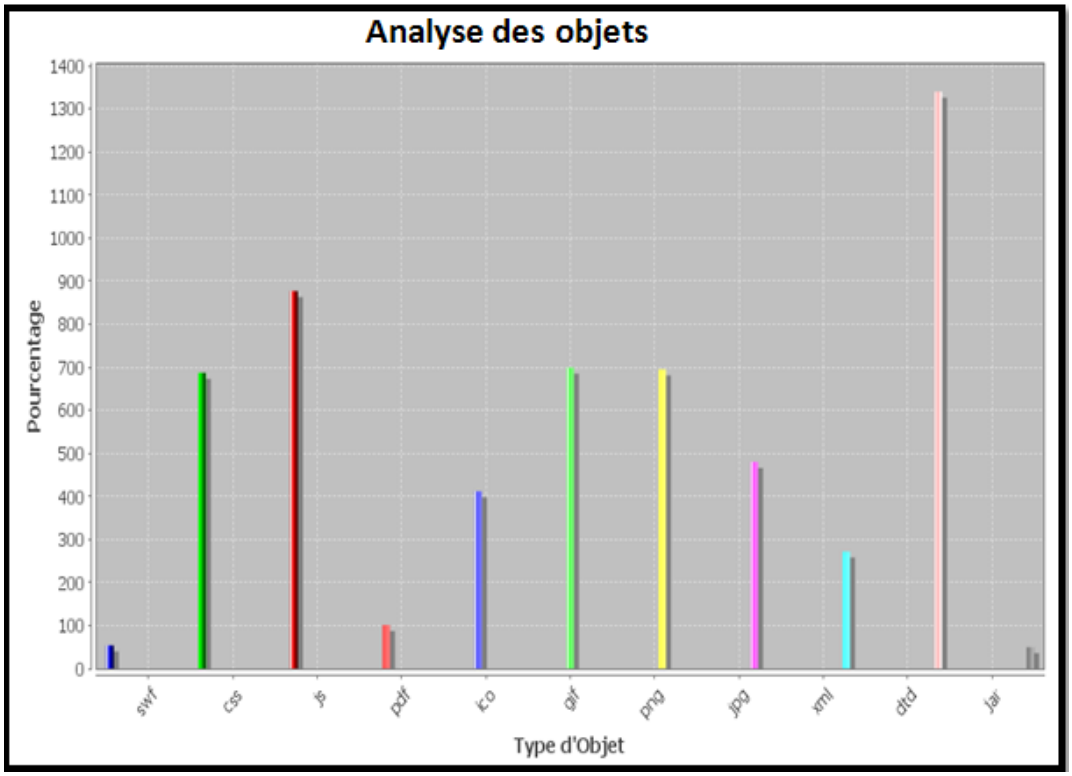


Figure V. 17: **Distribution des types d'objets.**

4.5 Visualisation par les outils existants

La Figure V.18 obtenue par Guess [37], montre la structure du graphe en cluster. Ceci justifie bien l'existence de nœud avec de forts degrés auxquels sont attachés des pages web de faibles degrés.

Cette figure est le graphe obtenu par le biais de l'outil Guess d'une capture qui contient **1513** pages et **6058** liens.

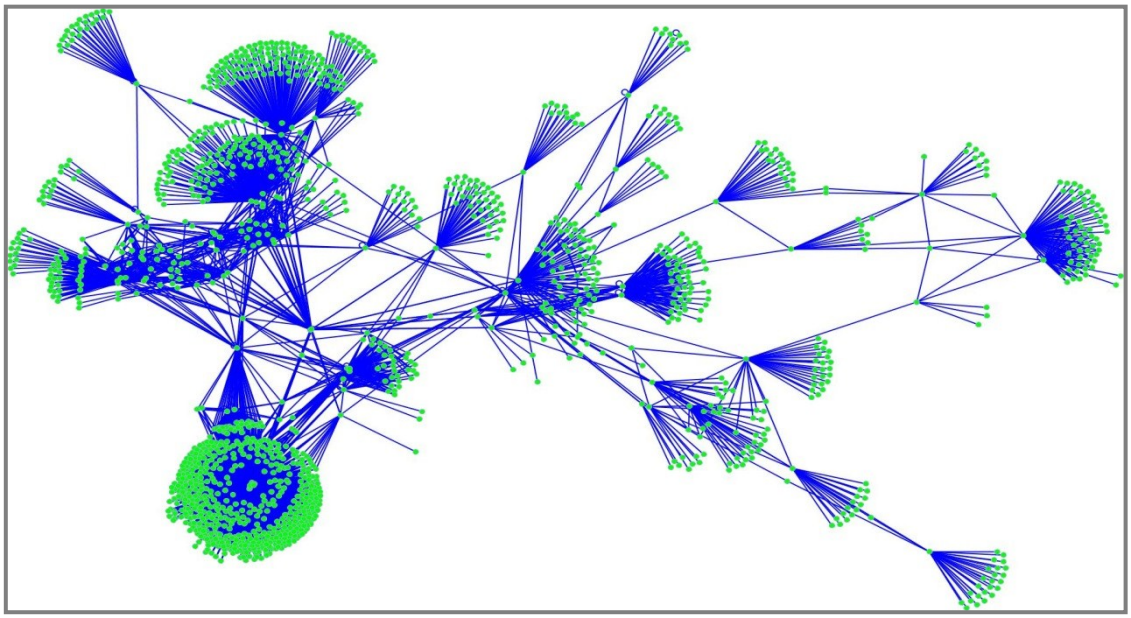


Figure V. 18*:* Visualisation en fonction du logiciel Guess (Clusters).

Nous avons exploité Guess pour faire un affichage selon la théorie du small world. Ainsi, sur la Figure V.19, où nous montrons que le small world est contenu dans le scale -free.

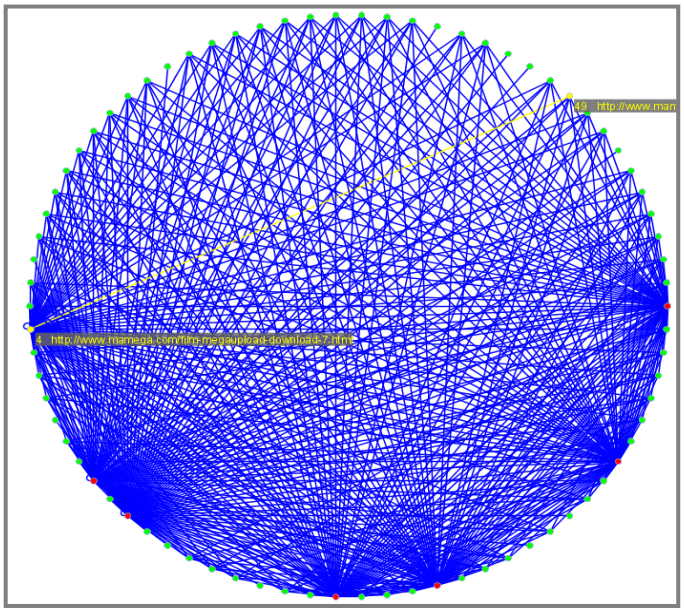


Figure V. 19*:* Visualisation en fonction du logiciel Guess.

5. Conclusion

Dans ce chapitre nous avions exposé l'ensemble de fenêtres qui permettent d'exploiter notre application. Nous avions exposé les motifs du choix du langage. Nous avons fait en sorte de donner une application souple et légère.

Notre objectif est bien atteint. Nous avions pu réaliser un crawler qui se faufile dans le web et qui peut récupérer efficacement des pages web. La partie récupération et indexation a été coûteuse est gourmande en temps. Nous avions pu gérer le heap afin d'éviter le blocage du programme. Les temps du calcul étaient pénalisants et c'est une chose commune à tous les crawlers.

En analysant l'ensemble de captures, nous avions ainsi joint la théorie de graphe d'interaction à savoir que la loi en puissance est la plus respectée. Ceci nous rassure et est réconfortant.

Nous avons pu extraire le texte à partir des pages Web crawlées et ce texte est transféré vers un notre module afin de concevoir un index Liens-Termes et Termes-Fréquences pour des travaux ultérieurs. L'index conçu indique les mots employés dans les pages traitées, le nombre de fois qu'un terme est apparu ainsi les liens indiquant la source de chaque terme. La collecte des informations était faite à partir des sites Algériens (une liste de 610 sites).

INTRODUCTION GÉNÉRALE

Il est apparu récemment que l'immense majorité des grands graphes rencontrés en pratique ont des propriétés bien particulières. Ceci a conduit la communauté scientifique à essayer d'identifier ces particularités et à les expliquer par l'intermédiaire de modèles de génération de graphes aléatoires ayant certaines propriétés. La compréhension de ces propriétés globales touche pourtant à des problématiques essentielles : la dynamique des interactions dans un réseau social ou un réseau informatique est par exemple liée à la problématique de la propagation des virus (informatiques ou biologiques), dans un réseau de distribution.

L'analyse, autrement dit le monitoring, est nécessaire lors que les systèmes deviennent larges. Ce procédé permet de détecter les anomalies et permet d'étendre les systèmes sous des bases théoriques solides. La théorie des grands graphes a dévoilé les propriétés cachées entre les composants des systèmes. Au début, on pensait que ces graphes étaient construits de façon aléatoire alors que les études actuelles montrent que ces relations suivent bien des lois de statistiques. Le problème qui survient par la suite est comment peut-on donner des algorithmes qui puissent simuler la réalité afin d'étudier les systèmes réels et en déceler les bugs.

Le World Wide Web fait partie de ces types de systèmes et il est connu pour sa vaste étendue à l'échelle internationale grâce au développement de l'Internet. A ses débuts, il ne comptait que quelques milliers de pages mais actuellement ils dépassent les 36 milliards sans compter le Web profond (deep Web).

Afin d'étudier le graphe du Web, il faut passer par la collecte des pages Web. Cette collection est appelée crawling ou exploration. Cette tâche est cruciale car elle nécessite des mécanismes de haut niveau pour que la collecte ne faille pas et que la composition du graphe soit correcte. Notons aussi que les moteurs de recherche incluent tous des crawlers.

Dans ce domaine, beaucoup d'études sont faites et c'est un processus qui tourne tant que le Web existe. Car il faut avoir toujours une étude aussi complète que possible, vu que le Web est en permanente extension.

Ce projet traite l'exploration, ou le crawling, comme sujet d'étude. L'objectif principal est donc : la réalisation propre d'un crawler qui puisse se faufiler dans le Web et retourner un maximum de pages Web sans fautes. L'étape, qui suit est une partie intégrante dans notre travail, est d'analyser le graphe réalisé afin d'en détecter les propriétés contenues et d'étudier la partie du monde Arabe dans le Web.

Notre travail porte aussi l'extraction de textes dans le sens où on tire pour chaque terme, la liste des pages Web où il apparait ainsi que sa fréquence d'apparition. Il s'agit de concevoir

un index inversé des termes afin de présenter aux utilisateurs un système de Recherche d'Information adaptatif et sophistiqué.

Le manuscrit que nous présentons est composé de cinq chapitres qui se définissent comme suit :

Le premier chapitre donne une introduction à la théorie des systemes complexes. Ainsi, il contient ses éléments exhaustifs. Nous présenterons les modèles de ces graphes ainsi que les algorithmes qui feignent leur réalité.

Le deuxième chapitre présente les détails du processus de crawling. Nous définissons qu'est ce qu'un crawler, son fonctionnment et sa stratégie de parcours du graphe du Web. Ce chapitre présente aussi les crawlers les plus populaires et dont les études académiques ont été dévoilées.

Le troisième chapitre sera destiné à la description des outils et des méthodes utilisées dans la Recherche d'Information. Afin de tirer parti la collecte des documents et la satisfaction des requêtes d'utilisateurs.

Le quatrième chapitre est consacré pour la conception que nous avons entreprise dans le but de réaliser notre crawler. Nous détaillerons le crawling, l'indexation, le parcours et les études que nous avons faites.

Le dernier chapitre présente en premier lieu notre logiciel en présentant son fenêtrage. En deuxième lieu, nous donnerons et commenterons les résultats de notre étude sur le graphe du Web obtenu.

En fin de ce transcrit, nous donnons une conclusion générale ainsi que les travaux futurs.

CONCLUSION GÉNÉRALE & PERSPECTIVES

Les premiers systèmes d'information n'étaient pas d'une grande taille de façon que leur monitoring fût simple et leur analyse fût aussi évidente. Avec les nouvelles technologies et l'expansion du Web ainsi avec l'effectif grandissant des utilisateurs, ce qui est convenu d'appeler passage à l'échelle, il a fallu trouver de nouveaux moyens pour s'acquérir et analyser l'information d'une manière concise.

Le Web actuel ne cesse de s'accroitre et par suite le nombre de pages Web est devenu explosif. On entend que le moteur de recherche Google indexe des milliards de pages Web, et malgré ceci il est loin de toucher à la totalité du Web. Ce moteur est au fait un analyseur qui avant d'indexer les pages, il faut qu'il les visite et les analyse. Ce processus de visite s'appelle crawling.

Dans ce domaine, nous avons défini la notion du crawling et nous en avons donné les objectifs et les mécanismes. Plus encore, nous avons présenté les crawlers les plus populaires et dont les études académiques ont été divulguées.

De notre part, nous avons réalisé notre propre crawler et analyseur de texte car nous étions toujours dépendants d'autres universités qui nous fournissaient des captures déjà étudiées et ainsi nous ne pouvions pas apporter des nouveautés leurs concernant. Notre crawler arrive aussi à récolter, analyser et animer les captures. Les captures et les analyses, que nous avons faites, ont montré que la façon avec laquelle nous avons entrepris les choses est correcte car nous avons trouvé les mêmes rapports théoriques.

La suite de ce travail présenté s'inscrit dans le contexte de la Recherche d'Information (RI) qui s'intéresse principalement à sélectionner à partir d'un ensemble de documents existants, ceux qui sont pertinents à une requête utilisateur. Afin d'y parvenir, l'une des tâches principales d'un Système de Recherche d'Information est l'indexation. Celle-ci consiste à construire des représentations simplifiées décrivant le contenu informationnel des documents et requêtes en vu de faciliter la recherche.

Dans le premier volet, nous avons pu faire une extraction des termes à partir des pages Web explorées. Une indexation par projection du texte sur une liste préétablie de toutes les captures indiquant le jour de collecte, le lien de la page crawlée et son texte avec leurs fréquences a été donnée. Nous nous sommes focalisés dans ce travail, sur le texte en langue Arabe. Cette contribution est la première, puisque toutes les captures faites par les autres universités sont en encodage Latin (Anglais, Français, …etc.)

Clicours.COM

Le deuxième volet, porte sur l'indexation des termes dans des documents sous format *.pdf*. Qui a pour but d'apporter une meilleure représentation du contenu des documents.

Les travaux futurs sont aussi un défi et consiste en les points suivants :

1. Multiplier les machines, c'est-à-dire, la création d'une grappe qui puisse faire lancer plus de threads.
2. Lancement d'un ensemble de threads parallèle par machine afin que les captures prennent moins de temps et soient plus grandes.
3. Tentative d'indexer et compresser les informations contenues dans les pages.
4. Introduire une sémantique qui permet de retrouver les pages Web selon un procédé intelligent.
5. Généraliser notre spectre de l'Algérie à l'Afrique en passant par le monde Arabe.

GLOSSAIRE

API (Application Programming Interface) :

Interface de programmation d'applications, contenant un ensemble de fonctions courantes de bas niveau**.**

Attachement préférentielle :

Un concept utilisé dans la construction des graphes, où un nouveau nœud a plus de chance d'être relié à un nœud qui a beaucoup de voisins par rapport à un autre qui en a moins.

Bande passante :

Cette expression est utilisée de manière courante pour désigner un débit exprimé en octets (ou Kilo, Méga, Giga-octet). Les services d'hébergement de site web proposent souvent une bande passante limitée (1Go, 5Go) utilisable librement pendant un mois. Au-delà de cette limite, soit le site est désactivé, soit la bande passante consommée en plus est payante.

Cache :

Espace de mémoire ou sont enregistrés temporairement des données, Le processeur (et seulement lui) y stocke les informations sur un espace en mémoire ou sur un disque dur destiné à enregistrer les pages web visitées (meilleur rapidité d'affichage lorsque l'on revient sur ces pages).

CERN :

Conseil Européen pour la Recherche Nucléaire. Il a gardé son acronyme mais s'appelle maintenant officiellement le Laboratoire Européen pour la Physique des Particules.

charset :

Encodage décrivant la nature des symboles et des caractères des langues utilisées dans la définition du texte dans le code HTML.

Coefficient de clustering :

Le coefficient de clustering (de clusterisation ou de regroupement) est une mesure de la connectivité d'un graphe non orienté.

Crawler :

Est un logiciel qui explore automatiquement un réseau (p2p, web), il est généralement conçu pour collecter les informations(le voisinage, les données ….etc.).

Crawler Axé :

Est un logiciel qui explore le réseau d'une manière automatique en cherchant et téléchargeant rien que des pages Web spécifiques à des sujets particuliers, contrairement aux autres crawlers qui parcourent le Web selon des stratégies traditionnelles.

Diamètre :

Le diamètre d'un graphe est le plus long chemin parmi l'ensemble des plus courts chemins pour tout couple de nœuds dans le graphe.

dir :

Attribut définit la direction de base d'un texte, de droite à gauche ou de gauche à droite dans une page HTML.

Distance moyenne :

La distance moyenne est la moyenne des longueurs des plus courts chemins pour tout couple de nœuds d'un graphe.

DNS :

domain name server ou domain name system, service essentiel de l'Internet assurant la conversion des noms de domaine en adresse IP.

Dynamicité :

Connexion / déconnexion (arrivée / départ) des machines (utilisateurs) dans le système.

GDF :

Geographic Data File, standard européen défini par le projet European Digital Road Map, spécifiant la création et la description de bases de données relatives aux routes. C'est une application de la géomatique.

Heap :

Tas en français, bloc de mémoire utilisé par un système exploitation pour les objets dynamique.

HTML :

Hyper Text Markup Language, est le langage universel utilisé pour communiquer sur le Web à base des balises.

http :

Hyper Text Transfert Protocol ou protocole de transfert hypertexte, protocole de transmission dédié aux clients et aux serveurs web. Facile à implanter car à un transfert de donnée est associé une connexion.

IDE :

Integrated Development Environment, réunissant tous les outils nécessaires à la création d'applications, aussi complexe qu'elles soient.

IR :

Information Retrieval, Recherche d'information. Un traitement mis à jour de tous les aspects de la conception et la mise en œuvre des systèmes de collecte, d'indexation et la recherche de documents.

lang :

Attribut utilisé dans le code HTML qui définit la langue de base du texte de la page Web.

Requête :

Mot, expression ou groupe de mots employés pour interroger un outil de recherche afin de localiser des fichiers sur le sujet recherché.

Resolver :

Programme permettant d'obtenir une adresse IP à partir d'un nom de domaine. Voir aussi résolution d'adresse.

Réseaux Scale-free :

Les réseaux scale-free ou sans échelle, sont des réseaux où les degrés sont très hétérogènes : la plupart des nœuds ont un faible degré, mais quelques-uns ont un fort degré.

Scalabilité :

La scalabilité ou passage à l'échelle est la capacité d'un système à évoluer en puissance, le plus souvent par ajout ou **remplacement** de composants.

SHA-1 :

Secure Hash Algorithm, algorithme de chiffrement, est une fonction de hachage cryptographique conçue par la *National Security Agency* des États-Unis (NSA).

URL :

Uniform Resource Locator. Sur le web, c'est la méthode d'accès à un document distant, créant ainsi par exemple un lien hypertexte, avec la syntaxe <type_connexion> :// <serveur> / <ressource> /... . Exemples : http:// agato.goglo.fr/ pub/.

Voisin :

Un nœud *u* est voisin de v s'il connaît son adresse IP. Il y a donc un lien overlay entre les deux nœuds.

World Wide Web :

La toile d'araignée mondiale des sites utilisant le protocole http. En français, on dit « la toile », ou le web (c'est le terme de loin le plus courant).

Web Invisible :

Aussi appelé web caché ou web profond, le web invisible est l'ensemble des documents diffusés par l'intermédiaire du web et qui ne sont ni lus, ni indexés par les moteurs de recherche.

XML :

Le langage XML (eXtended Markup Language), est un langage de format de document. Il dérive de SGML (Standard Generalized Markup Language).

SOMMAIRE

Chapitre II : Etat de l'art des Crawlers

Chapitre III : Etat de l'art sur la Recherche d'Information

Chapitre IV : Conception

clicours.com

Chapitre V : Implémentation

Clicours.COM

TABLE DES FIGURES

Chapitre I : Introduction aux Graphes d'Interaction

Chapitre II: Etat de l'Art des Crawlers

Chapitre III: Etat de l'Art sur la Recherche d'Information

Chapitre IV : Conception

Chapitre V : Implémentation

TABLE DES EQUATIONS

Chapitre I : Introduction aux Graphes d'Interaction

Chapitre II: Etat de l'Art des Crawlers

LISTE DES TABLEAUX

Chapitre I : Introduction aux Graphes d'Interaction

Chapitre III: Etat de l'Art sur la Recherche d'Information

Chapitre IV : Conception

TABLE DES CODES

Chapitre IV : Conception

TABLE DES ALGORITHMES

Chapitre IV : Conception

BIBLIOGRAPHIE

[1] Emanuelle Lebhar : « *Algorithmes de routage et modèles aléatoire pour les graphes petits mondes* » ; Thèse de doctorat, Ecole Normale Supérieure de Lyon, Laboratoire de l'Informatique du Parallélisme, Lyon, France, Décembre 2005.

[2] I. Ferrer, R.Cancho, R.V.Solé : « *The small-world of human language* » ; Proceedings of the Royal Society of London Biol. Sc, pp. 19-22.268:2261–2265, 2001. Cité pages(s) 19, 20, 21, 22.

[3] Rushed Kanawati : « *Détection de communautés dans les grands graphes d'interactions (multiplexes)» ;* Rapport d'un état de l'art. 2013.

[4] E. Fleury, J.-L. Guillaume, C. Robardet et A. Scherrer : « *Loi de puissance et caractérisation des réseaux dynamiques » ;* conf´erenciers `a Infocom, Hyatt Regency hotel `a Miami, USA. 2007.

[5] Ramesh Govindan, Hongsuda Tangmunarunkit : « *Heuristics for internet map discovery* » ; In Proceedings of the 2000 IEEE INFOCOM Conference, pp. 1371–1380, Tel Aviv, Israel, March 2000. Cité pages(s) 16, 20, 21, 55.

[6] A.-L.Barabasi, H.Jeong, Z.Néda, E.Ravasz, A.Schubert T.Vicsek : « *Evolution of the social network of scientific collaborations* » ; In Physica A 311, pp. 590–614, 2002. Cité pages(s) 18, 20, 21, 22.

[7] Lada A.Adamic : « *The small world web* » ; In Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libraries ; pp. 443–452, Springer-Verlag, 1999. Cité pages(s) 17, 21, 22, 81.

[8] D.Watts, S.Strogatz : « *Collective dynamics of small-word networks* » ; Nature, 393, pp. 440-442, 1998.

[9] S.Deerwester , S.T.Dumais, T.K.Landauer, G.W. Furnas and R.A. Harshman. « *Indexing by latent semantic analysis* » Journal of the Society for Information Scence, 41(6),391-407,1990.

[10] S.Chakrabarti , B.Dom, D.Gibson, J.Kleinberg, P.Raghavan and S.Rajagopalan : *« Mining information networks through spectral methods »*. In preparation, IBM Umaden Research Center, 1997.

[11] K.B.Petersen, M .S.Pedersen, *« The matrix cookbook» ;* version 2008.

[12] Spyros Voulgaris, Anne-Marie Kermarrec, Laurent Massoulié, Maarten Van Steen « *Exploiting Semantic Proximity in Peer-to-Peer Content Searching* » ; In Proceedings the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS), pp. 238-243, Suzhou, China, May 26-28, 2004.

[13] A.Crespo, H.Garcia-Molina : « *Semantic overlay networks for P2P systems* » ; Technical report, Stanford University, January 2003.

[14] G.Salton: « *The SMART Retrieval System* », Prentice Hall, 1971.

[15] S.Brin, L.Page: « *The anatomy of a large-scale hypertextual Web search engine* »,

WWW7, Brisbane, Australia, 1998.

[16] G.Salton, Mc. M. Gill, « *Introduction to Modern Information Retrieval* », McGraw-Hill, 1983.

[17] R.Baeza-Yates, B.Ribeiro-Neto: « *Modern Information Retrieval* », Addison- Wesley, 1999.

[18] Will, T: « *Introduction to the Singular Value Decomposition.* »; Davidson College*;* 1999.

[19] G.W.Furnas, Landauert., L.Gomez, S.Dumais: « *The vocabulary problem in human-system communication* », Communications of the Association for Computing Machinery, vol. 30, 1987, p. 964–971.

[20] Anthony Ventresque : « *Une mesure de similarité sémantique utilisant des résultats de psychologie »* ; Laboratoire d'Informatique de Nantes Atlantique (LINA) ; Manuscrit auteur, publié dans "COnférence en Recherche d'Infomations et Applications - CORIA 2006, Lyon : France (2006)".

[21] IHADJADENE Madjid. « *Usages des moteurs de recherche. In Journée d'Etude ADBS,*

Optimiser l'accès à l'information, une opportunité pour les langages documentaires ? », Paris, 2007.

[22] Edwards, McCurley, K. S., and Tomlin, J. A. *"An adaptive model for optimizing performance of an incremental web crawler". In Proceedings of the Tenth Conference on WorW Wide Web (Hong Kong: Elsevier Science):* 106-m. 45p-6op. 2001.

[23] Castillo, Carlos (2004). Effectue Web Crawling (Ph.D. thesis). University of Chile. Retrieved 2010-08-03

[24] O.A.McBryan, « *Tools For Taming The Web. In Proceedings Of The First International Conference On World Wide Web* », Chicago USA 1993.

cicours.com

[25] B.Pinkerton, « *Finding What People Want: Experiences With The Crawler. In Proceedings Of The First International Conference On World Wide Web »*, Chicago USA 1993.

[26] D.Eichmann,« *The Rbse Spider – Balancing Effective Search Against Web Load, Proceedings Of The First International Conference On World Wide Web* », Chicago USA1993.

[27] Gulli, A.; Signorini, A. *"The indexable web is more than 11.5 billion pages". Special interest tracks and posters of the 14th international conference on World Wide Web. ACM Press., pp. 902-903. 2005.*

[28] Lawrence, Steve; c. Lee Giles *"Accessibility of information on the web". Nature 400(6740): 107. Bibcode i999Natur.400..i07L.1038/21987. pmid 10428673. 1999-07-08.*

[29] Cho, J.; Garcia-Molina, H.; Page, L. *(1998-04)."Efficient Crawling Through URL Ordering". Seventh International World-Wide Web Conference. Brisbane, Australia. Retrieved 2009-03-23*

[30] Marc Najork and Janet L. Wiener. *Breadth-first crawling yields high-guality pages. In Proceedings of the Tenth Conference on World Wide Web, pages 114-118, Hong Kong, May 2001. Elsevier Science*

[31] Boldi, Paolo; Bruno Codenotti, Massimo Santini, Sebastiano Vigna *(2004). "UbiCrawler: a scalable fully distributed Web crawler". Software: Practice and Experience 34 (8): 711— 726. doi:10.1002/spe.587. Retrieved 2009-03-23.*

[32] Boldi, Paolo; Massimo Santini, Sebastiano Vigna *(2004). "Do Your Worst to Make the Best: Paradoxical Effects in PageRank Incremental Computations" and Models for the Web-Graph, pp. 168-180. Retrieved 2009-03-23.*

[33] Baeza-Yates, R., Castillo, C, Marin, M. and Rodriguez, *A. (2005). Crawling a Country: Better Strategies than Breadth-First for Web Page Ordering. In Proceedings of the Industrial and Practical Experience track of the 14th conference on World Wide Web, pages 864-872, Chiba, Japan. ACM Press.*

[34] Shervin Daneshpajouh, Mojtaba Mohammadi Nasiri, Mohammad Ghodsi. "*A Fast Community Based Algorithm for Generating Crawler Seeds Set", In proceeding of 4th International Conference on Web Information Systems and Technologies (WEBIST-2008), Funchal, Portugal, May 2008.*

[35] [Menczer, F. *(1997). ARACHNID: "Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery. In D. Fisher, ed., Machine Learning": Proceedings of the 14th International Conference (ICML97). Morgan Kaufmann*

[36] Menczer, F. and Belew, R.K. *(1998). "Adaptive Information Agents in Distributed Textual Environments. In K. Sycara and M. Wooldridge (eds.) "Proc. 2nd Intl. Conf. on Autonomous Agents (Agents '98). ACM Press*

[37] Chakrabarti, S., van den Berg, M., and Dom, B. *(1999). "Focused crawling: a new approach to topic-specific web resource discovery". Computer Networks, 31(11-16):1623-1640*

[38] Pinkerton, B. *(1994). "Finding what people want: Experiences with the WebCrawler". In Proceedings of the First World Wide Web Conference, Geneva, Switzerland*

[39] Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. L, and Gori, M. *(2000)." Focused crawling using context graphs." In Proceedings of 26th International Conference on Very Large Databases (VLDB), pages 527-534, Cairo, Egypt*

[40] Pant, Gautam; Srinivasan, Padmini; Menczer, Filippo *(2004). "Crawling the Web". In Levene, Mark; Poulovassilis, Alexandra. Web Dynamics: Adapting to Change in Content, Size, Topology and Use. Springer. pp. 153-178. ISBN 978-3-540-40676-1. Retrieved 2009-03-22.*

[41] Cothey, Viv *(2004). "Web-crawling reliability". Joumal of the American Society for Information Science and Technology55 (14).*

[42] Jr, e. g coffman; ztien uu, Richard R. Weber *(1998). "Optimal robot scheduling for Web search engines". Journal of Scheduling 1(1): 15-29. doi:10.1002/(SICni099-1425(199806)1:1<15::AID-JOS3>3.0.CO:2-K*

[43] Cho, j. and Garcia-Molina, H. *(2003). "Effective page refresh policies for web crawlers." ACM Transactions on Database Systems, 28(4)*

[44] Jr,E. G. Coffman; Zhen Liu, Richard R. Weber *(1998). "Optimal robot scheduling for Web search engines". Journal of Scheduling 1 (1): 15-29.doi:10.1002/(SICni099-1425(199806)1:K15::AID-JOS3>3.0.CO;2-K*

[45] Cho, Junghoo; Hector Garcia-Molina *(2003). "Estimating frequency of change". ACM Trans. Intérêt Technol. 3 (3): 256- 290.doi:io.ii45/857i66.857i7o. Retrieved 2009-03-22.*

[46] Ipeirotis, P., Ntoulas, A., Cho, J., Gravano, L. *(2005) "Modeling and managing content changes in text databases". In Proceedings of the 21st IEEE International Conference on Data Engineering, pages 606-617, April 2005, Tokyo.*

[47] Koster, M. *(1995). "Robots in the web: threat or treat?" Connexions, 9(4)*

[48] Koster, M. *(1996). "A standard for robot exclusion."*

Clicours.COM

[49] Peshave, M. and Dezhgosha,K. *(2006). "How search engines works". Technical report, of University Plaza*

[50] Cho, J. and Garcia-Molina, H. *(2003). "Effective page refresh policies for web crawlers." ACM Transactions on Database Systems.*

[51] Baeza-Yates, R. and Castillo, C. *(2002). "Balancing volume, quality and freshness In Web crawling." In Soft Computing Systems - Design, Management and Applications, pages 565-572, Santiago, Chile. IOS Press Amsterdam.*

[52] Heydon, Allan; Najork, Marc *(1999-06-26) (PDF). Mercatori A Scalable, Extensible Web Crawler, http://www.cindoc.csic.es/cybemietrics/pdf/68.pdf. Retrieved 2009-03-22*

[53] Dill, S., Kumar, R., Mccurley, K. S., Rajagopalan, S., Sivakumar, D., and Tomkins, A. *(2002)." Self-similarity in the web." ACM Trans. Inter. Tech., 2(3):205-223.*

[54] web crawling ethics revisited*: Cost, privacy and denial of service". Journal of the American Society for Information Science and Technology. 2006 . http://www.scit.wlv.ac.uk/%7Ecm1993/papersA/Veb_Crawling_Ethics_preprint.*

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

[56] Sergey Brin and Lawrence Page, "*The Anatomy of a Large-Scale Hypertextual" Web Search Engine, 1998.*

[57] Michalis Faloutsos, Petros Faloutsos, et Christos Faloutsos. « *On power-Iaw relationships of the internet topology". In SIGCOMM, pages 251-262, 1999.*

[58] d.m. CvetkovTc, m. Boob, et h. Sachs, "*spectra of Graphs. Académie press", 1979.*

[59] Soon-Hyung Yook, Hawoong Jeong, et Albert-Laszlo Barabasi. "*Modeling the internet's large-scale topology." PNAS, 99(21) 113382-13386, 2002.*

[60] R. Albert, H. Jeong, et A.-L. Barabasi. *"Diameter of the world wide web". Nature, 401 :13o-131, 1999.*

[61] Albert-Lâszlô Barabâsi, Réka Albert, et Hawoong Jeong. "*Scalefree characteristics of random networks : the topology of the World Wide Web." Physica A : Statistical Mechanics and its Applications, 281169-77, 2000.*

[62] Jon M. kleinberg, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Andrew Tomkins, et Janet Wiener. "*The web as a graph : measurements,*

models, and methods." In Proceedings of the 5th International Conference on Computing and Combinatorics(COCOON), July 1999.

[63] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, et Andrew Tomkins. "*Crawling the Web for emerging cyber-communities." Computer Networks (Amsterdam, Netherlands : 1999), 31(11-16) 1481-1493, 1999.*

[64] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, et Janet Wiener. "*Graph structure in the web." In Proceedings of the Ninth International World Wide Web Conference. Elsevier, 2000.*

[65] Ravi Kumar S., Prabhakar Raghavan, Sridhar Rajagopalan, D. Sivakumar, Andrew Tomkins, et Eli Upfal. "*The web as a graph." In Proceedings of the Symposium on Principles of Database Systems, pages 1-10, 2000.*

[66] Dellil C., Mekki H. et ZEKRI L.: « A*nalyse des systèmes large échelle : Réalisation d'un crawler du web», Université d'Oran es-sénia, mémoire de fin études D'Ingénieur d'État en Informatique. 2011*

[67] jon M. Kleinberg. "*Authoritative sources in a hyperlinked environment". J. ACM, 46(5) :6o4-632, 1999.*

[68] Chakrabarti S., B. Dom, D. Gibson, J. Kleinberg, S.R Kumar, P. Raghavan, S. Rajagopalan, et A. Tomkins. "*Hypersearching the web". Scientific American, 280 -.54-60, June 1999*

[69] Chakrabarti s., B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, et S. Rajagopalan. « *Automatic resource list compilation by analyzing hyperlink structure and associated text". In Proceedings of the 7th International World Wide Web Conference, pages 65-74, 1998.*

[70] Krishna Bharat et Monika R. Henzinger. "*Improved algorithms for topic distillation in a hyperlinked environment." In Proceedings of SIGIR- 98, 21st ACM International Conference on Research and Development in Information Retrieval, pages 104-m, Melbourne, AU, 1998.*

[71] Lawrence Page, Sergey Brin, Rajeev Motwani, et Terry Winograd. *"The pagerank citation ranking : Bringing order to the web". Technical report, Stanford Digital Library Technologies Project, 1998.*

[72] Sergey Brin, Rajeev Motwani, Lawrence Page, et Terry Winograd. *"What can you do with a web in your pocket?" Data Engineering Bulletin, 21(2) 37-47,1998.*

[73] Jon M. Kleinberg. "*Authoritative sources in a hyperlinked environment." J. ACM, 46(5) -.604-612, 1999.*

[74] Paul Alexandru Chirita, Daniel Olmedilla, etWolfgang Nejdl. "*Finding related hubs and authorities." In Proceedings of the ist Latin-American Web Congress, Santiago, Chile, 2003.*

[75] Andrew y. Ng, Alice x. Zheng, et Michael I. Jordan. "*Stable algorithms for link analysis". In Proceedings of the 24th Annual Intl. ACM SIGIR Conference. ACM, 2001.*

[76] Lempel r. et S. Moran. « *The stochastic approach for linkstructure analysis (SALSA) and the TKC effect" j-COMP-NET-AMSTERDAM, 33(1-6) 387-401, June 2000.*

[77] Megague K., Zekri L. et Senouci M. : « *évaluation d'utilité et d'efficacité des informations dans les moteurs de recherche »,Conférence sur les TIC SNTIC'2013. Université d'USTO-MB Algérie. Juillet 2013.*

[78] LANCASTER, F.W., "*Information Retrieval Systems: Characteristics, Testing and Evaluation", Wiley, New York (1968)*

[79] WINOGRAD, T., "*Understanding Natural Language", Edinburgh University Press, Edinburgh (1972).*

[80] MINSKY, M, "*Semantic Information Processing", MIT Press, Cambridge, Massachusetts (1968).*

[81] Christopher D.Manning, Prabhakar Raghavan Hinrich Schütze :*«An Introduction to Information Retrieval»*; Cambridge UP; 2009.

[82] BAR-HILLEL, Y., Language and Information. "*Selected Essays on their Theory and Application", Addison-Wesley, Reading, Massachusetts (1964)*

[83] Djoerd Hiemstra : « *Information Retrieval Models* », Published in: Goker, A., and Davies, J. Information Retrieval: Searching in the 21st Century. John Wiley and Sons, Ltd., ISBN-13: 978-0470027622, November 2009*

[84] BARBER, A.S., BARRACLOUGH, E.D. and GRAY, W.A. "*On-line information retrieval as a scientist's tool", Information Storage and Retrieval, 9, 429-44- (1973).*

[85] Vana Kalogeraki, Dimitrios Gunopulos, Demetrios Zeinalipour-Yazti : « *A Local Search Mechanism for Peer-to-Peer Networks* » ; November 4-9, 2002.

[86] M.Najork, J.L.Wiener, « *Breadth-first Crawling Yields High-Quality Page. In Proceedings Of The Tenth International Conference On World Wide Web »*, HongKong 2001.

WEBOGRAPHIE

[w1] http://forum.webrankinfo.com

[w2] http/fr.m.wikipedia.org/wiki/Loi_de_puissance

[w3] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#lancaster79

[w4] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#meadow92

[w5] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#soergel85

[w6] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#vickery87

[w7] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#cjvr79

[w8] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#tague91

[w9] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#blcr92

[w10] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#marcus91

[w11] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#frby92

[w12] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#cooper88

[w13] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#fxkl88

[w14] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#marcus93

[w15] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#furnas83

[w16] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#coupe91

[w17] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#stalon83

[w18] http://people.ischool.berkeley.edu/~buckland/papers/analysis/node6.html#belkin93

[w19] http://www.projrt-plume.org/fiche/gephi.

[w20] http://graphexploration.cond.org.

[w21] http://www.cs.utexas.edu/users/EWD/

RÉSUMÉ

L'exploration, ou le crawling en anglais, est le processus de collecte et d'analyse des pages Web. Les objectifs de cette exploration est de déceler les propriétés existantes entre les sommets du graphe du Web et l'indexation des contenus, en extrayant les mots et les objets contenus dans ces pages.

Dans ce domaine, nous allons définir la notion du crawling et nous en allons donner les objectifs et les mécanismes. Afin de donner une connaissance plus concise, nous présenterons les crawlers les plus connus.

Afin de devenir indépendant, nous avons décidé de monter un crawler propre à nous. Nous montrerons les différentes étapes réalisées et l'ensemble des expérimentations que nous avions menées afin de valider notre crawler. Effectivement, nous prouvons que notre analyse est correcte car nous avons utilisé des outils d'analyse qui ont illustré que nous avons trouvé les mêmes résultats. Nous montrons aussi que nous avons rejoint les mêmes résultats théoriques des graphes d'interaction.

Les systèmes de recherche d'information reposent sur l'indexation par les mots-clés pour représenter le contenu des documents. La première étape pour construire des indexes, est de collecter et générer leurs contenus. Dans ce travail, nous avons mis l'accent sur l'extraction du texte à partir de notre crawler et notamment en langue Arabe.

Mots clés : crawling, systèmes large échelle, page Web, capture, graphe d'interaction, propriétés, loi de puissance, Recherche d'Information, collecte, indexation, extraction.

ABSTRACT

Exploration, or crawling, is the process of collecting and analyzing of Web pages. The objectives of this exploration are the detection of the existing properties between the vertices of the Web graph and the indexing of its content, by extracting the words and the objects contained in these pages.

In this field, we will define the notion of the crawling and we give its objectives and mechanisms. In order to give more concise idea, we will present the most known crawlers.

In order to become independent, we have decided create our own crawler. We will show the various steps that we have taken in the designing process of our crawler and we give in this thesis the results of our experimentation in order to validate our crawler. Indeed, we will show that our analysis is correct because we have used tools for analysis which showed that we have found the same results. We will also show that we joined the same theoretical results in complex systems theory.

Information Retrieval systems are based on indexation by the keywords to represent the documents contents. The first step to make indexes is to collect and generate content. In this work, we've focused on text extraction from our proper crawler especially in Arabic.

Keywords: crawling, large scale systems, Web page, snapshot, graph interaction, properties, power law, Information Retrieval, gathering, indexing, extraction.

RÉSUMÉ

L'exploration, ou le crawling en anglais, est le processus de collecte et d'analyse des pages Web. Les objectifs de cette exploration est de déceler les propriétés existantes entre les sommets du graphe du Web et l'indexation des contenus, en extrayant les mots et les objets contenus dans ces pages.

Dans ce domaine, nous allons définir la notion du crawling et nous en allons donner les objectifs et les mécanismes. Afin de donner une connaissance plus concise, nous présenterons les crawlers les plus connus.

Afin de devenir indépendant, nous avons décidé de monter un crawler propre à nous. Nous montrerons les différentes étapes réalisées et l'ensemble des expérimentations que nous avions menées afin de valider notre crawler. Effectivement, nous prouvons que notre analyse est correcte car nous avons utilisé des outils d'analyse qui ont illustré que nous avons trouvé les mêmes résultats. Nous montrons aussi que nous avons rejoint les mêmes résultats théoriques des graphes d'interaction.

Les systèmes de recherche d'information reposent sur l'indexation par les mots-clés pour représenter le contenu des documents. La première étape pour construire des indexes, est de collecter et générer leurs contenus. Dans ce travail, nous avons mis l'accent sur l'extraction du texte à partir de notre crawler et notamment en langue Arabe.

Mots clés :

Crawling; Systèmes Large Echelle; Page Web; Capture; Graphe D'interaction; Propriétés; Loi De Puissance; Recherche d'Information; Collecte; Indexation; Extraction.

ABSTRACT

Exploration, or crawling, is the process of collecting and analyzing of Web pages. The objectives of this exploration are the detection of the existing properties between the vertices of the Web graph and the indexing of its content, by extracting the words and the objects contained in these pages.

In this field, we will define the notion of the crawling and we give its objectives and mechanisms. In order to give more concise idea, we will present the most known crawlers.

In order to become independent, we have decided create our own crawler. We will show the various steps that we have taken in the designing process of our crawler and we give in this thesis the results of our experimentation in order to validate our crawler. Indeed, we will show that our analysis is correct because we have used tools for analysis which showed that we have found the same results. We will also show that we joined the same theoretical results in complex systems theory.

Information Retrieval systems are based on indexation by the keywords to represent the documents contents. The first step to make indexes is to collect and generate content. In this work, we've focused on text extraction from our proper crawler especially in Arabic.

Keywords:

Crawling; Large Scale Systems; Web Page; Snapshot; Graph Interaction; Properties; Power Law; Information Retrieval; Gathering; Indexing; Extraction.