Spygraph : un robot d'exploration léger dédié à l'analyse de graphes d'hyperliens

Robert Viseur[1]

1. FWEG - Université de Mons
Service de Technologies de l'Information et de la Communication
17, place Warocqué, B-7000 Mons
robert.viseur@umons.ac.be

RÉSUMÉ. Spygraph est un programme écrit en Python permettant l'exploration d'un ensemble de sites web de manière à en extraire les hyperliens. Facilement installable, configurable et exécutable, il outille l'exploration d'un écosystème local au travers des sites web des acteurs qui le composent. Basé sur une structure de données simplifiée en SQLite, il permet de configurer l'exportation des hyperliens vers différents outils utiles à l'analyse tels que LibreOffice.org Calc (tableur) ou Gephi (analyse de graphe). Dès lors, il facilite la découverte de nouveaux acteurs au travers des domaines liés et de l'analyse de leurs relations en ligne. Spygraph se positionne donc comme une solution légère, adaptée aux graphes de petite taille, en complémentarité avec des solutions plus complexes comme Hyphe.

ABSTRACT. Spygraph is a program written in Python allowing the exploration of a set of websites in order to extract hyperlinks. Easily installable, configurable and executable, it enables the exploration of a local ecosystem through the websites of the organisations that make it up. Based on a simplified data structure in SQLite, it allows to configure the export of hyperlinks to different tools useful for the analysis such as LibreOffice.org Calc (spreadsheet) or Gephi (graph analysis). It therefore facilitates the discovery of new actors through linked domains and the analysis of their online relationships. Spygraph is thus positioned as a light solution, adapted to small graphs, in complementarity with more complex solutions like Hyphe.

MOTS-CLÉS : analyse de graphe, robot d'exploration, Gephi.

KEYWORDS: graph analysis, crawler, Gephi.

1. Contexte et présentation générale

Spygraph a été développé dans le cadre du projet Interreg FabricAr3v[1]. L'objectif était de disposer d'un outil léger, basé sur des composants libres et *open source*, permettant l'exploration d'un écosystème local, transfrontalier, composé d'acteurs économiques actifs dans la fabrication numérique, au travers des hyperliens reliant leurs sites web, et en complément d'approches quantitatives (p. ex. sondages) ou qualitatives (p. ex. entretiens). À cette fin, des métriques d'analyse de graphe ont été utilisées avec le logiciel dédié Gephi. (cf. Hansen *et al.*, 2010 pour

[1] Cf. https://fabricar3v.eu/.

une présentation des méthodologies d'analyse de graphe et Figure 1 pour un exemple de graphe tiré du projet FabricAr3v).

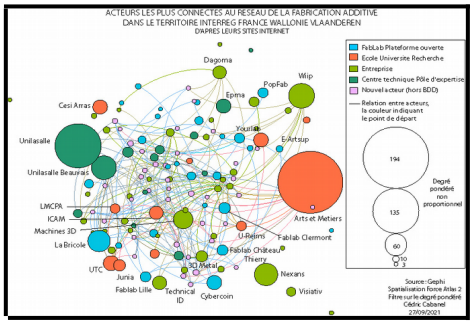

Figure 1. Résultat de l'analyse de graphe

Spygraph est donc un programme dédié à l'exploration d'un ensemble de sites web (robot) et à la production d'une base de données d'hyperliens. L'exportation dans un format de fichier adapté permet d'ensuite réaliser une analyse des relations entre sites web à l'aide d'un logiciel libre commun comme LibreOffice.org Calc (tableur) ou spécialisé comme Gephi[2] (analyse de graphe). Spygraph est développé en Python et s'appuie sur une base de données SQLite. Il utilise des bibliothèques courantes telles que urlib (manipulation et lecture des URLs) et Beautiful Soup[3] (lecture des documents HTML) de manière à en faciliter l'installation, l'extension et l'usage. L'outil se veut léger et flexible ; il n'ambitionne pas de remplacer des solutions plus complexes, adaptées aux plus grands graphes, comme Hyphe[4].

2. Utilisation du programme

Le script principal, actuellement testé sous Ubuntu Linux, doit être lancé en ligne de commande et accepte des paramètres d'entrée. L'avancée du processus d'exploration peut être suivie dans un terminal de type Bash sous Ubuntu Linux. Un cycle d'exploration complet peut être lancé à l'aide d'un script Bash :

```
#!/bin/bash
python3 run.py init <monprojet>
python3 run.py crawl inside <monprojet>
python3 run.py export csv <monprojet>
python3 run.py export graph <monprojet>
```

Le programme exporte un fichier DOT ainsi qu'un fichier CSV. Ce dernier peut être importé dans un classeur LibreOffice.org Calc dès lors que le nombre de lignes n'excède pas les limites du format Open Document Format, soit 1.048.576 lignes dans le cas des fichiers portant l'extension « `ods` »[5].

La configuration du programme se fait, d'une part, lors de l'exécution de la ligne de commande (cf. Tableau 1), d'autre part, au sein d'un fichier de configuration. La configuration se fait au sein d'un fichier textuel (`run.cfg`), compatible avec le

---

2 Cf. https://gephi.org/.
3 Cf. https://beautiful-soup-4.readthedocs.io/.
4 Cf. https://github.com/medialab/hyphe.
5 Cf. https://wiki.documentfoundation.org/Faq/Calc/022.

module Python Config Parser, permettant de paramétrer la profondeur de l'exploration, le nombre maximal d'URIs traitées avant arrêt, le *timeout*, ainsi que les requêtes SQL utilisées par défaut pour l'exportation des données.

Tableau 1. Paramètres de ligne de commande

`init`	Création de la base de données et initialisation de la table « urls_init » avec la liste d'URLs d'amorçage.	
`crawl`	Exécution de l'exploration sur base de la liste d'URLs.	
	`inside`	Exploration des URLs associées aux domaines fournis dans la liste d'amorçage uniquement.
	`outside`	Exploration de toutes les URLs trouvées (mode divergent).
`export`	Exportation des données issues de la base de données une fois l'exploration terminée.	
	`csv`	Exportation des liens entre pages au format CSV.
	`graph`	Exportation des liens entre pages au format DOT.

Sur un ordinateur Dell E6540, équipé d'un disque dur SSD, d'un processeur Intel Core i7-4810MQ (2,80GHz) et de 16GB de RAM, tournant sous Ubuntu Linux, connecté à un réseau universitaire (fournisseur d'accès Belnet), la vitesse d'exploration, en fonctionnement *monothread*, est de l'ordre de 75 URLs par minute. Sur une table d'URLs comportant plus d'un million d'entrées, l'exécution d'une requête de sélection impliquant des jointures et/ou des regroupements n'excède pas 5 à 10 secondes.

3. Structure de base de données

La base de données est composée de trois tables (cf. Tableau 2). La première contient la liste des URLs d'amorçage. Chaque URL est associée au *fully qualified domain name* (FQDN) et au nom de domaine calculés (p. ex. *commons.wikimedia.org* et *wikimedia.org*). La seconde contient la liste des URLs découvertes. La troisième contient la liste des liens entre pages, identifiés par les pages sources et destinations, associées à leurs domaines respectifs. *Stricto sensu* la table « `urls` » contient des URIs incluant les URLs mais aussi des schémas d'URIs (comme « *mailto* », « *tel* » ou « *javascript* »). Chaque entrée y est associée à un champ « `iscrawled` », indiquant si l'URI a été visitée, un champ « `isignored` », indiquant si l'URI a été ignorée (cas des schémas autres que les schémas d'URL), et un champ « `iserror` », indiquant si l'ouverture ou la lecture de la page ont déclenché une erreur. Ces champs permettent l'identification d'éventuels problèmes lors de l'exploration. Ils ont par exemple permis d'identifier, sur les premières versions du robot, des problèmes d'instabilité réseau et de vérification de certificats SSL.

Tableau 2. Structure de la base de données

urls_init	**urls**	**links**
url fulldomain domainname	url fulldomain domainname iscrawled iserror isignored	hyperlink_from hyperlink_to fulldomain_from fulldomain_to domainname_from domainname_to

L'utilisation de la base de données SQLite permet une navigation aisée dans les résultats de l'exploration que l'on travaille sous Windows, macOS ou GNU/Linux. L'analyse des résultats de l'exploration, la sélection des données ainsi que le filtrage des données lors de l'exportation peuvent en effet être réalisés en interrogeant directement la base de données en utilisant la console SQL du logiciel libre DB Brower for SQLite[6]. À l'aide du SQL, il est ainsi possible de rapidement découvrir de nouveaux sites et de dresser des statistiques sur base des résultats de l'exploration : nombre d'URLs utilisées en amorçage, nombre d'URIs traitées, nombre d'URLs explorées, nombre de liens extraits, nombre de liens extraits sans doublon, nombre d'URIs identifiées… Les novices en SQL peuvent se rabattre sur un outil de tableur, alimenté par le fichier CSV exporté, tel que Microsoft Excel ou LibreOffice.org Calc (p. ex. calcul du nombre d'URL par domaine unique et détection des domaines découverts lors de l'exploration).

4. Perspectives

Premièrement, la lecture des documents pour en extraire les hyperliens pourrait être complétée par une extraction de terminologie. Cela permettrait, d'une part, la génération automatique d'un nuage de mots-clefs pour un ensemble de domaines sélectionnés par requêtes (cf. Viseur, 2014a pour un retour d'expérience avec des outils NLP *open source*), d'autre part, le filtrage des URLS, lors de l'exportation, sur base d'une liste de mots-clefs. Deuxièmement, le contenu des pages pourrait être utilisé pour créer un index plein texte à l'aide d'une technologie *open source* telle que Lucene[7] ou un de ses portages (p. ex. PyLucene) (cf. Viseur, 2012, et Viseur, 2014b, pour des exemples). Troisièmement, et plutôt que de privilégier la couverture maximale d'un ensemble de sites web prédéfinis, l'objectif étant de découvrir les acteurs en interaction sur un territoire donné au travers des liens entre leurs sites web respectifs, le robot pourrait intégrer des techniques de « *focused crawling* » afin de réduire la taille du corpus d'URLs (cf. Micarelli & Gasparetti, 2007 pour une introduction) sans altérer les qualités exploratoires du robot.

Bibliographie

Hansen D., Shneiderman B. Smith M.A. (2010). *Analyzing social media networks with NodeXL: Insights from a connected world*. Morgan Kaufmann.

Micarelli A. & Gasparetti F. (2007). Adaptive focused crawling. In *The adaptive web* (pp. 231-262). Springer, Berlin, Heidelberg.

Viseur R. (2014a). Automating the Shaping of Metadata Extracted from a Company Website with Open Source Tools. *International Journal of Advanced Computer Science and Applications*.

Viseur R. (2014b). Initial Results from the Study of the Open Source Sector in Belgium. *International Symposium on Open Collaboration*, Berlin, Germany.

Viseur R. (2012). Create a Specialized Search Engine: The Case of an RSS Search Engine. *International Conference on Data Technologies and Applications*, Rome, Italie.

[6] Cf. https://sqlitebrowser.org/.
[7] Cf. https://lucene.apache.org/.