WEM : Web Mining
Laboratoire n°1
Crawling, indexation et recherche de pages Web

28.03.2020

Objectifs

Ce laboratoire a pour but d'implémenter un web crawler parcourant une collection de pages web de votre choix, puis d'utiliser le logiciel *Apache Solr*[1] pour l'indexation et la recherche.

Les points étudiés dans ce laboratoire seront :

- Crawling de pages web (en utilisant la librairie *crawler4j*[2])
- Construction d'un index avec le logiciel *Solr*
- Implémentation d'une fonction de recherche

Durée

- 6 périodes. A rendre le vendredi **20.03.2020** à **8h30** au plus tard.

Références

- Cours «Web Mining» de Laura Raileanu
- Livre « Web Data Mining » de Bing Liu
- Livre « Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage » de Zdravko Markov et Daniel T. Larose
  Livre « Mining the Social Web » de Matthew Russell
- Livre «Text Data Management and Analysis » de ChengXiang Zhai et Sean Massung

Donnée

Le laboratoire à réaliser en *Java* se déroulera en quatre parties. Dans un premier temps, vous crawlerez un site web de votre choix à l'aide de la librairie *crawler4j*, on utilisera une instance par défaut de *Solr* pour l'indexation. Dans la seconde partie, nous allons configurer plus précisément l'instance de *Solr* utilisée par rapport à nos données. Ensuite on utilisera l'index créé dans la seconde partie pour mettre en place une fonction de recherche. La dernière partie consiste à répondre quelques questions théoriques.

[1] http://lucene.apache.org/solr/
[2] https://github.com/yasserg/crawler4j

1. Crawler

Pour cette partie nous allons utiliser un « core » par défaut de *Solr*. Il n'y a pas vraiment d'installation pour *Solr*, il suffit de le dé-zipper et d'exécuter la commande `bin/solr start` et vous pourrez ensuite accéder à la console web par l'adresse suivante : http://localhost:8983/solr/#/
Vous pourrez ensuite créer votre premier « core » avec la commande `bin/solr create -c <name>`
Vous noterez l'affichage dans la console de l'avertissement « WARNING: Using _default configset. Data driven schema functionality is enabled by default, which is NOT RECOMMENDED for production use» nous reviendrons dessus dans la seconde partie de ce laboratoire.
L'utilisation de la librairie *crawler4j* repose principalement sur l'implémentation par vos soins d'une classe étendant *edu.uci.ics.crawler4j.crawler.WebCrawler*, vous devrez en particulier implémenter les 2 méthodes suivantes :

public boolean shouldVisit(Page referringPage, WebURL url)

Pour chaque lien rencontré par le crawler durant sa visite, il demandera à cette méthode si la page doit être visitée (téléchargée). A vous de faire en sorte que les ressources vraisemblablement non-supportées (non-textuelles) ou inutiles ne soient pas visitées. **Vous limiterez aussi la visite uniquement au domaine ciblé.**

public void visit(Page page)

Cette méthode sera appelée par le crawler pour chaque page visitée. Vous pouvez ici décider de limiter l'indexation uniquement aux formats supportés, tous les cas ne pouvant pas être évités avec la méthode précédente. Cette méthode mettra en forme le contenu de la page en vue de son indexation.

Une fois le contenu des pages visitées indexé, vous pouvez afficher quelques statistiques à partir de la console de Solr, la Fig. 1 vous montre le nombre de documents présents dans l'index d'un des cores.

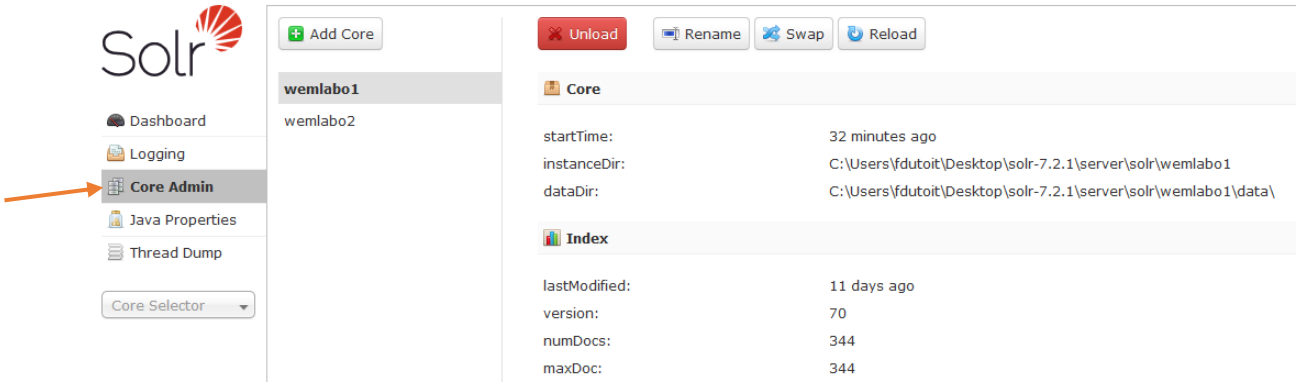

Figure 1 - Information sur les cores d'une instance de Solr

Quelques remarques/précisions supplémentaires

- Veuillez noter l'existence de la librairie *solrj*[3] qui vous aidera certainement à faire le lien entre votre crawler et *Solr*.
- *Crawler4j* lance plusieurs threads en parallèle, cela implique certainement qu'il vous faudra prendre en compte la concurrence dans votre programme.
- Le crawler reste en attente un certain temps en fin de crawling (jusqu'à 30 secondes). Ceci est un comportement normal.

[3] https://lucene.apache.org/solr/guide/7_2/using-solrj.html#common-configuration-options

- La classe *edu.uci.ics.crawler4j.crawler.CrawlConfig* permet de configurer le crawler, vous noterez la méthode *setMaxPagesToFetch()* permettant de limiter le nombre de pages à visiter, cela peut être utile pour limiter le temps de crawl.

2. Indexation spécialisée

L'index créé dans la première partie comporte plusieurs champs auto-générés, vous noterez principalement sur la Fig. 2 que pour chaque champ créé, *Solr* va auto-générer un second champ (par exemple title et title_str). Pour configurer un core de *Solr*, cela se passe dans le dossier `<dossier installation de solr>/server/solr/<corename>` dans lequel on trouve les fichiers suivants :

- **core.properties**
  En ajoutant le paramètre update.autoCreateFields=false cela permettra de désactiver la génération automatique des champs

- **conf/managed-schema**
  On trouvera dans ce fichier XML tous les types de champs définis `<fieldType>` ainsi que les champs définis pour ce core `<field>`

- **conf/solrconfig.xml**
  Permet la configuration des requestHandler, les points d'entrées permettant d'effectuer des recherches sur le contenu de l'index

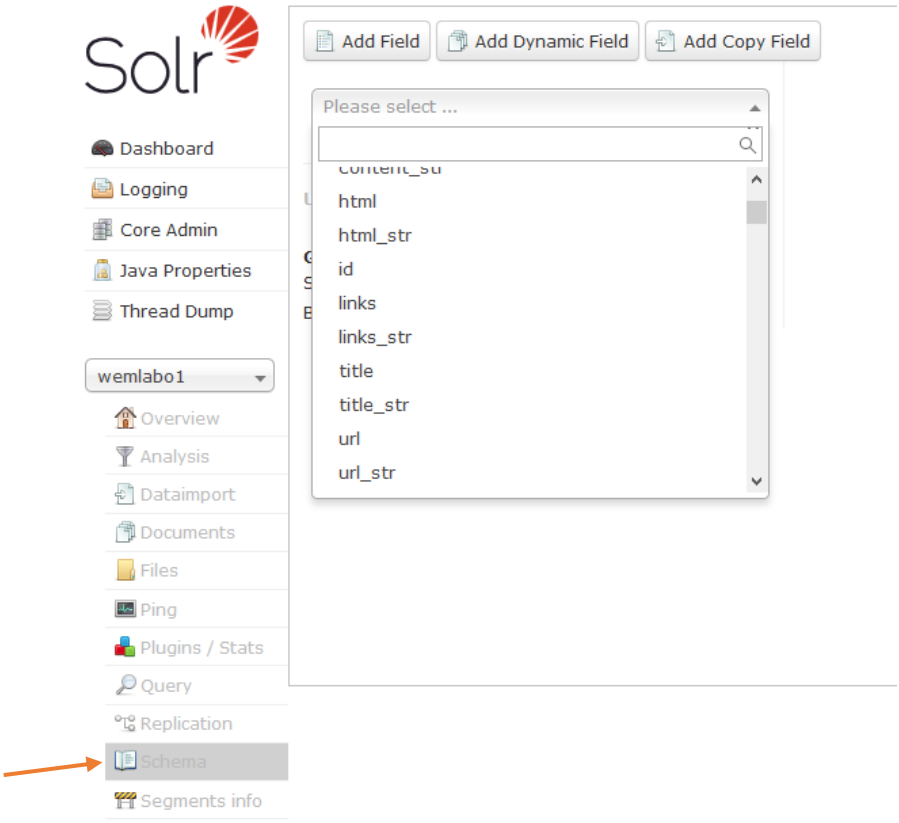

Figure 2 - Liste des champs présents dans un core

On souhaite à présent configurer notre index beaucoup plus finement, vous devrez créer les champs associés pour tous les éléments principaux composant une page web. Veuillez lister et justifier vos choix ainsi que les paramètres utilisés dans votre rapport.

De plus, comme le montre la Fig. 3, sur une page HTML certains éléments peuvent être mis en avant, vous adapterez votre crawler pour qu'il récupère ces éléments et les indexe dans leurs propres champs. Veuillez préciser dans votre rapport quel(s) élément(s) vous avez décidé d'indexer dans leurs propres champs et comment l'avez-vous réalisé.

Figure 3 – Sur certaines pages web, une catégorisation du contenu est proposée

La grande majorité des pages HTML fournissent des données structurées afin de mieux comprendre le contenu de la page. On vous demande également d'adapter votre crawler afin qui récupère ces données structurées (normes schema.org ou Open Graph à choix).

Ces données ne seront pas forcément utilisées pour l'indexation mais seront à fournir lors de la recherche de documents. Vous documenterez également quel(s) élément(s) vous avez décidé de stocker et comment.

```
<script type="application/ld+json">
{
   "@context":"https://schema.org",
   "@type":"Article",
   "name":"Haute Ecole d'ingénierie et de gestion du canton de Vaud",
   "description":"La Haute École d'Ingénierie et de Gestion du Canton de Vaud a été créée
   à la suite du regroupement de l'École d'ingénieurs du canton de Vaud (EIVD) et de la
   Haute école de gestion du canton de Vaud (HEG-VD), le 1er août 2004. Elle propose
   différentes formations intégrées au système de Bologne : Bachelors of Science, Masters
   of Science et plusieurs programmes exécutifs en formation continue.",
   "author":{
         "@type":"Organization",
         "name":"Contributeurs aux projets de Wikimédia"
   }
   "datePublished":"2006-07-27T00:09:58Z",
   "dateModified":"2020-01-19T22:52:19Z",
   "image":"https://upload.wikimedia.org/wikipedia/commons/f/fe/Locator_Map_Kanton_Waadt.p
   ng"
}
</script>
```

Figure 4 –Exemple de données structurées au format JSON-LD

Quelques remarques/précisions supplémentaires

- Veuillez noter l'existence de la librairie *jsoup*[4] qui vous aidera certainement à récupérer du contenu spécifique dans une page au format HTML.
- Il ne faut pas oublier de recharger (bouton reload visible sur la Fig. 1) le core pour que les modifications apportées dans les fichiers de configuration soient prises en compte.

[4] https://jsoup.org/

3. Recherche

A partir de l'index créé à la partie 2, veuillez utiliser l'interface de *Solr* pour réaliser des recherches, la Fig. 4 vous indique l'emplacement de l'outil permettant d'effectuer des recherches. Veuillez effectuer une recherche par défaut (q=*/*) et une recherche avec un mot qui se trouve dans l'index. Que constatez-vous, veuillez donner des explications dans votre rapport.

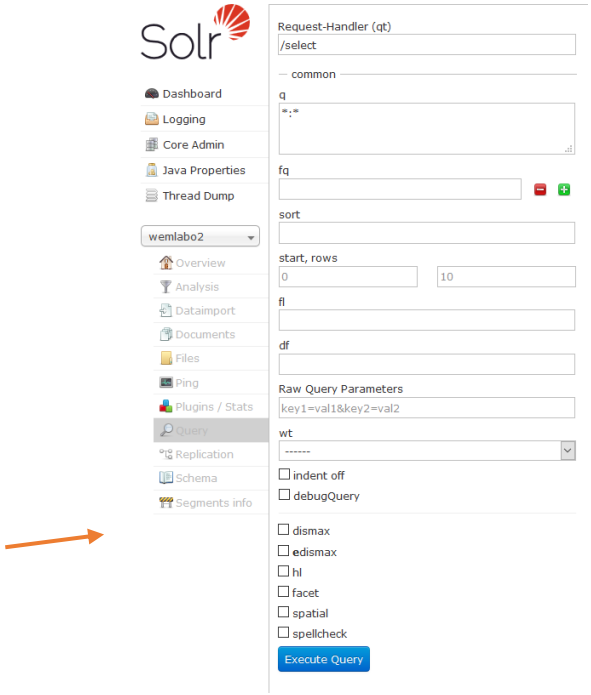

Figure 4 – Outil de recherche mise à disposition par l'interface web de Solr

Vous ajouterez ensuite une fonctionnalité de recherche à votre programme *Java*, celle-ci permettra d'effectuer une recherche dans votre index et d'afficher les résultats obtenus ainsi que leur score (pertinence). Veuillez mettre en place une recherche qui privilégiera la recherche dans le titre d'une page et dans les champs « mis en avant » (cf. Fig. 3) par rapport à tout le contenu de la page.

4. Questions théoriques

4.1. Veuillez expliquer quelle stratégie il faut adopter pour indexer des pages dans plusieurs langues (chaque page est composée d'une seule langue, mais le corpus comporte des pages dans plusieurs langues). A quoi faut-il faire particulièrement attention ? Veuillez expliquer la démarche que vous proposez.

4.2. *Solr* permet par défaut de faire de la recherche floue (fuzzy search). Veuillez expliquer de quoi il s'agit et comment Solr l'a implémenté. Certains prénoms peuvent avoir beaucoup de variation orthographiques (par exemple Caitlin : Caitilin, Caitlen, Caitlinn, Caitlyn, Caitlyne, Caitlynn, Cateline, Catelinn, Catelyn, Catelynn, Catlain, Catlin, Catline, Catlyn, Catlynn, Kaitlin, Kaitlinn, Kaitlyn, Kaitlynn, Katelin, Katelyn, Katelynn, etc). Est-il possible d'utiliser, tout en gardant une bonne performance, la recherche floue mise à disposition par Solr pour faire une recherche prenant en compte de telles variations ? Sinon quelle(s) alternative(s) voyez-vous, veuillez justifier votre réponse.

Rendu/Evaluation

Vous remettrez sur *Moodle* un zip contenant les sources, les libraires utilisées, les éventuels fichiers d'entrée, etc. Vous ajouterez en plus dans votre rendu les fichiers de configuration utilisés pour vos cores *Solr* et un petit rapport dans lequel vous discuterez du fonctionnement de votre programme, de vos choix d'implémentation et répondrez aux questions posées.

Vous pouvez discuter entre les groupes mais il est strictement interdit d'échanger du code.

Adresse E-Mail de l'assistant : antoine.rochat@heig-vd.ch

Bonne chance !