Projet Solr

Un moteur de recherche sur les personnages Marvel et DC

Alexane Jouglar

Sorbonne Université

Dans le cadre de notre enseignement d'indexation sémantique et de recherche d'information, nous avons travaillé au développement d'un moteur de recherche. Ce travail se divisait en trois grandes parties. Il a fallu tout d'abord choisir un corpus, l'indexer puis mettre en place le moteur de recherche grâce à Velocity. C'est dans cet ordre que nous parlerons de ce travail.

Le choix du corpus

Le corpus utilisé renseigne les caractéristiques de 16 376 personnages Marvel et 6 896 personnages DC Comics. Les deux ensembles ont été fusionnés au sein d'un même fichier renommé marvel_dc_data.csv et enregistré au sein du dossier ExempleDocs de Solr. Pour chaque personnage, nous avons douze caractéristiques.

Les données viennent des wiki respectivement dédiés aux personnages Marvel et DC Comics, et sont librement disponibles depuis un repository GitHub (https://github.com/fivethirtyeight/data/tree/master/comic-characters). La licence est la suivante : CC-BY-4.0 License.

A l'aide d'expressions régulières, certaines colonnes ont été modifiées. L'url de chaque document a été complétée avec l'ajout d'un http://... qui n'existait pas dans le fichier original. La colonne *alive* est devenue booléenne : si le personnage est vivant, il est marqué *true*, sinon *false*.

Table 1: Colonnes du fichier csv : caractéristiques pour chaque document d'après le README du corpus.[a]

page$_id$	L'identifiant de la page du personnage au sein du wiki dédié au monde en question
name	Le nom du personnage
urlslug	L'url qui mène vers la page wiki du personnage
ID	si l'identité du personnage est secrète, publique ou les deux
align	si le personnage fait partie des méchants ou des gentils
eye	la couleur des yeux
hair	la couleur des cheveux
sex	le sexe du personnage
gsm	si le personnage fait partie d'une minorité sexuelle
alive	si le personnage est vivant ou décédé
appearances	le nombre d'apparition du personnage dans les comics
year	l'année d'apparition
id	L'identifiant unique donné par défaut pour ce travail

[a]de ces douze colonnes, seule une n'est pas d'origine : l'id qui a été généré ultérieurement pour obtenir un identifiant unique pour chaque personnage. Cet identifiant peut être incrémenté pour ajouter des personnages au fichier.

Indexation des données

La première étape consistait en la création d'une collection MarvelDC en solr. Pour ce faire, en Linux, il suffit de se rendre dans le dossier bin de la plateforme Solr, d'y ouvrir un terminal et d'exécuter la commande suivante :

./solr create -c MarvelDC

Il s'agit alors de modifier le schéma d'indexation présent dans le fichier xml managed-schema de la collection MarvelDC et d'indiquer le type de chaque champ. Nous récapitulons dans le tableau ci-dessus ces informations.

Pour indéxer les documents à la collection créée, nous nous rendons dans le dossier ExampleDocs (au sein d'Example), où l'on a précédemment ajouté le fichier marvel_dc_data.csv, y ouvrons un terminal et exécutons la commande suivante :

java -Dtype=text/csv -Dcommit=yes -Dc=MarvelDC -jar post.jar mar-

Table 2: Type pour chaque champ d'un document.

align	text_general
alive	booleans
appearances	plong
eye	text_general
genre	string
gsm	text_general
hair	text_general
id	string
name	text_general
page_id	pint
secrecy	text_general
sex	text_general
urlslug	text_general

vel_dc_data.csv

Enfin, pour chaque champ donné précédemment, nous indiquons s'il est indexé (c'est-à-dire s'il va pouvoir être cherché par notre moteur de recherche), s'il est stocké, s'il peut contenir plusieurs valeurs et s'il est requis. Tous sont stockés, aucun ne contient des valeurs multiples, seul l'id est intrinsèquement requis. La caractéristique "stored" est dite vraie pour tous les champs. Seule le caractère indexé ou non va changer d'un champ à l'autre. Le tableau 3 reprend les choix réalisés pour cette caractéristique.

Lancement du moteur de recherche avec Velocity

La troisième étape repose sur l'utilisation de Velocity. Pour ce faire, nous utilisons le fichier Velocity d'ores-et-déjà formé et présent dans la collection technoProducts précédemment réalisé en cours. Nous copions ce fichier tel quel dans notre serveur au sein du dossier conf. Plusieurs modifications sont nécessaires pour que le moteur de recherche soit fonctionnel.

Nous proposons des facettes et nous les inscrivons dans le fichier solrconfig.xml.

Table 3: Les champs et leur indexation.

CHAMP	Indexé
name	oui
id	oui
align	oui
alive	non
appearances	non
genre	oui
eye	oui
hair	oui
sex	oui
gsm	oui
urlslug	non
year	oui
page_id	non
secrecy	oui

Table 4: Champs utilisés pour chaque type de facette.

3*Facettes de champ	genre
	align
	sex
2*Facettes d'intervalles	appearances
	year
Facettes pivot	alive

Le tableau ci-dessous reprend les facettes utilisées et leur type.

Nous inscrivons dans le fichier solrconfig.xml de notre collection les balises *lib*, *QueryResponseWriter*, et *resquestHandler* tel qu'indiqué dans le td5.

<codecFactory class="solr.SchemaCodecFactory"/>

<schemaFactory class="ManagedIndexSchemaFactory"> <bool name="mutable">true</bool> <str name="managedSchemaResourceName">managed-schema</str> </schemaFactory>

<lib dir="$solr.install.dir:../../../../contrib/velocity/lib" regex=".*jar" />

```
<span class="general">
<div class="result-title"><span class="title">#field('name')</span></div>
<div><span class="id">Id: #field('id')</span></div>
<br/>
<div><u>URL:</u> <span class="wiki"><a href=#field('urlslug') target=_blank>page
wiki du personnage</a><span></div>
<div><big><u>Genre:</u> <i><b>#field('genre')</i></b></big></div>
<div><u>Secrecy of the character's identity:</u> #field('secrecy')</div>
<div><u>Align:</u> #field('align')</div>
<div><u>Year:</u> <b>#field('year')</b></div>
<br/>
<div>Is the character alive? <i><b>#field('alive')</b></i></div>
<div>How many times did the character appear? #field('appearances')</div>
<br/>
<div><span class="chara">Other characteristics : </span></div>
 <div class="row">
  <div class="column"><u>Sex:</u> #field('sex')</div>
  <div class="column"><u>Hair:</u> #field('hair')</div>
  <div class="column"><u>Eye:</u> #field('eye')</div>
</div>
<span>
```

Figure 1. Elements visibles dans les résultats d'une recherche.

<lib dir="$solr.install.dir:../../../../dist/" regex="solr-velocity-.*jar" />
<queryResponseWriter name="velocity" class="solr.VelocityResponseWriter"
startup="lazy"> <str name="template.base.dir">$velocity.template.base.dir:</str>
</queryResponseWriter>

<requestHandler name="/browse" class="solr.SearchHandler"> [...] </requestHandler>

Nous choisissons ensuite les champs qui seront visibles pour un document donné dans les résultats d'une recherche. On les renseigne dans le fichier product_doc.vm présent dans Velocity tel que visible dans la figure 1.

Par ailleurs, il est nécessaire de supprimer dans les fichiers adéquats les éléments de l'interface que nous ne souhaitons pas voir.

La dernière étape repose sur la modification du style. En html, nous mettons

```
<!-- Highlighting defaults -->
<str name="hl">on</str>
<str name="hl.fl">name align secrecy alive appearances sex genre</str>
<str name="hl.preserveMulti">true</str>
<str name="hl.encoder">html</str>
<str name="hl.simple.pre">&lt;span class="highlight"&gt;</str>
<str name="hl.simple.post">&lt;/span&gt;</str>
```

Figure 2. Elements visibles dans les résultats d'une recherche.

```
.highlight {
  color: blue;
  background-color: aliceblue;
  padding-top: 3pt;
  padding-bottom: 1pt;
  border-radius: 10pt;
}
```

Figure 3. Style donnés aux éléments recherchés.

des éléments en gras ou en italique, ou nous soulignons le nom des champs. Ensuite, nous faisons la modification du main.css. Cette étape n'est pas indispensable, elle sert simplement à personnaliser l'environnement. Le tableau 5 reprend les changements réalisés et la figure 2 indique que nous souhaitons mettre en relief les éléments d'une recherche dans les champs indiqués en hl.fl. La figure 3 montre le style que l'on souhaite donner à ces éléments recherchés.

Conclusion

Dans ce rapport nous avons présenté le travail réalisé pour le module d'indexation sémantique. Ce travail consistait en la construction d'un moteur de recherche à partir d'une liste de documents. Nous avons choisi un fichier csv portant sur les personnages des univers Marvel et DC. Ce travail nous a permis d'en apprendre davantage sur le fonctionnement des moteurs de recherche et sur les possibilités qu'ils offrent.

Table 5: Changements réalisés dans le main.css.

Element concerné	Modification
Les angles	ajout d'un "border-radius: 10px"
navigator h2	suppression du background
Field facets	ajout d'un background mauve
Police générale	Lucida Sans Regular, Lucida Grande, Lucida Sans Unicode, Verdana, sans-serif
Police des résultats	Courier New, Courier, monospace
ID	width: 100px ; height: 100px ; background-color: rgb (255, 255, 255); color: rgb(255,0,157); border-color: #ff004c; border-style: solid; border-width: 1pt; border-radius: 10pt; padding: 5px; font-size: 15pt; opacity: 0.6; transition: 0.3s;
Lien de la page wiki	et son hover : font-size: 20pt;
Nom du personnage	font-size: 20pt; color: ae00ff;
Autres caractéristiques	Mise en relief du titre + mise en colonnes grâce à display:flex
Background des résultats	changement de la couleur .result-document:nth-child(2n+1) background-color:rgb(235,176,176);