Documentation

SEO Echo

Julien Cordat-Auclair

13 août 2018

Table des matières

1 Présentation générale

`SEO Echo` est un service permettant de soumettre des recommandations à un utilisateur vis-à-vis du contenu de sa page web pour qu'il puisse optimiser le référencement de celle-ci en fonction d'une requête donnée. Il y a donc 2 paramètres que l'utilisateur doit fournir : l'URL de sa page et des mots-clés qui constituent la requête pour laquelle l'utilisateur souhaite voir sa page mieux placée dans les résultats de recherche Google.

Afin de mettre en place ce projet, l'idée est d'utiliser l'algorithme d'apprentissage machine `Echo` de manière à extraire les termes qui permettent de différencier un site bien référencé d'un site mal référencé pour une requête donnée. Ce sont ces termes qui seront proposés à l'utilisateur. 3 étapes distinctes apparaissent alors :

- 1ère étape : le web scraping. Lors de cette étape, on récupère des données afin de pouvoir faire fonctionner l'algorithme `Echo`. Ces données correspondent au contenu présent dans les balises HTML (que l'on choisit) des n pages web les mieux référencées pour la requête donnée, l'entier n pouvant être modifié.
- 2ème étape : l'apprentissage machine. On fournit les données précédemment récupérées à `Echo` qui retourne les termes qui permettent d'être mieux référencé (via `EchoBT`) mais aussi une estimation de la position de la page de l'utilisateur (via `EchoPos`) et un taux de performance permettant de rendre compte de la pertinence des résultats de chacun des deux algorithmes précédents (via `EchoV`).
- 3ème étape : l'UX design. C'est l'optimisation de l'affichage des résultats renvoyés par `Echo` à partir des données fournies.

Avant de détailler chacune des étapes présentées ci-dessus, voici un exemple concret de l'action de SEO Echo. Le premier tableau représente les données fournies par l'utilisateur et le second correspond aux résultats renvoyés par `Echo` :

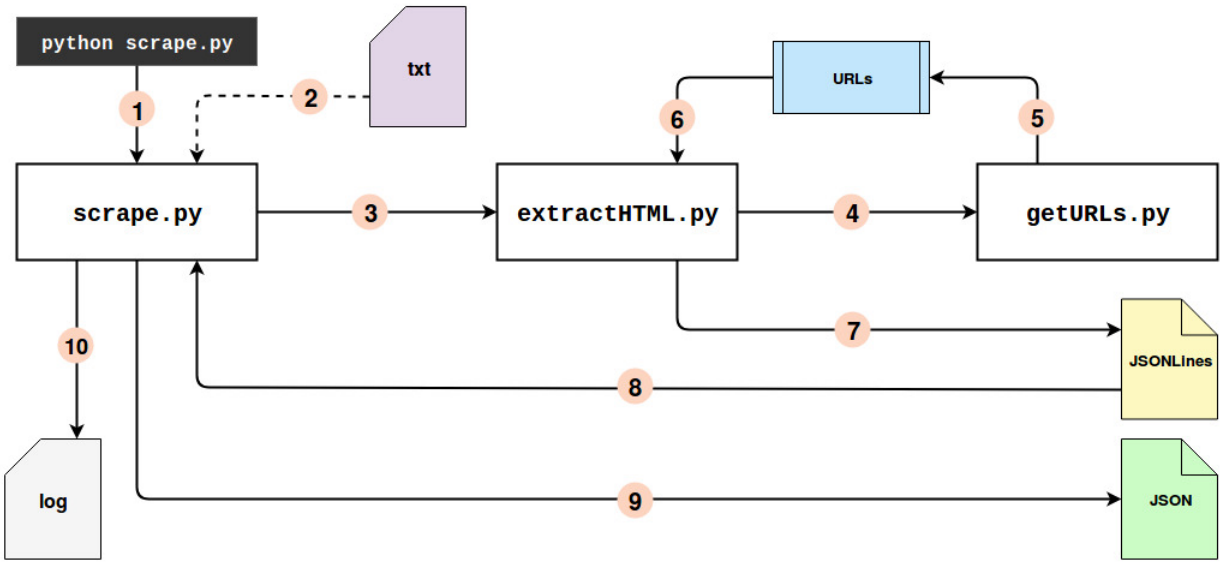
URL de l'utilisateur	http://com-et-net.com
Mots-clés	stratégie digitale grenoble

`EchoBT`	référencement_title, agence_title, stratégie_alt, isère_strong ...
`EchoPos`	14
`EchoV`	0.76, 0.63

2 Le web scraping

pour plus d'information, se référer aux commentaires dans le code

Voici un schéma représentant l'architecture de cette première étape :



Description des étapes

- 1 : le script *scrape.py* est exécuté via la commande tapée dans le shell
- 2 : les paramètres contenus dans le fichier texte sont enregistrés (URL utilisateur, nombre de résultats, degré de pertinence et mots-clés)
- 3 : le script *scrape.py* exécute le script *extractHTML.py* avec les paramètres récupérés
- 4 : le script *extractHTML.py* exécute le script *getURLs.py* avec les paramètres récupérés
- 5 : ce dernier renvoie une liste des n (nombre de résultats) premières URLs s'affichant suite à la requête (composée des mots-clés) donnée
- 6 : la liste d'URLs est renvoyée au script *extractHTML.py*
- 7 : le script *extractHTML.py* génère un fichier JSONLines comportant les données HTML des balises spécifiées, un indice de pertinence et une position pour chacune des URLs
- 8 : le fichier JSONLines est lu par le script *scrape.py*
- 9 : le script *scrape.py* génère un fichier JSON où chaque groupe de données associé à une URL est trié par sa position
- 10 : le script *scrape.py* génère aussi un fichier de log comportant toutes les informations liées au déroulement du processus

`scrape.py` : c'est le script principal qui permet de lancer le processus de web scraping.

- **log** : le script génère un fichier de log qui permet de suivre le déroulement du processus et ainsi de repérer les éventuelles erreurs pouvant survenir. Il est écrasé à chaque exécution et se trouve dans le dossier */CrawlGooglePython/bin/log*.

- **paramètres** : le script est également en mesure de lire un fichier texte en interprétant son contenu de la manière suivante :

1ère ligne	URL de l'utilisateur
2ème ligne	nombre de pages à étudier
3ème ligne	degré de pertinence
lignes suivantes	mots-clés

- **lancement du processus** : les paramètres précédemment récupérés sont utilisés pour appeler le script `extractHTML.py` qui permet d'initialiser le web crawling (voir plus loin).

- **tri des résultats** : une fois les résultats du web crawling obtenus, le script est capable de trier le fichier `JSONLines` que `Scrapy` génère et contenant les données `HTML` (désordonnées car `Scrapy` est asynchrone) en fonction de la position de chacune des pages étudiées.

- **génération d'un fichier de sortie** : un nouveau fichier `JSON` (et non `JSONLines`) est créé de manière à ce qu'il puisse être lu plus tard par le script `PHP` (`PHP` ne lit pas les fichiers `JSONLines`). Il se trouve dans le dossier */CrawlGooglePython/bin/searchEngine/output* et se nomme *output.json*.

`extractHTML.py` : c'est le script capable d'extraire les données `HTML` des pages web. Il utilise le framework `Scrapy`.

- **cleanHTML** : fonction chargée de nettoyer le contenu extrait des pages web car `Echo` n'est pas capable de traiter les données avec des accents, des signes de ponctuation ou encore des espaces vides.

- **cleanURL** : fonction chargée de nettoyer une URL. Elle est utilisée lorsque l'on souhaite récupérer les mots-clés présents dans le lien de la page et séparés par des anti-slashs.

- **lancement de la récupération d'URLs** : afin de collecter les données des pages web, encore faut-il avoir ces pages à disposition. C'est pourquoi le script permet d'exécuter `getURLs.py` (voir plus loin) grâce aux paramètres précédemment transmis par `scrape.py`.

- **start_requests** : fonction chargée d'attribuer à chaque URL récupérée sa position dans la page de recherche Google ainsi qu'un indice de pertinence. Par exemple, si le degré de pertinence vaut 30, alors les 30 premières pages auront un indice de pertinence de 1 et les autres de 2. On attribue par ailleurs à la page de l'utilisateur (elle aussi étudiée) la position 0 et un indice de pertinence de 0.

- **parse** : fonction principale de ce script chargée de récupérer le contenu des balises spécifiées pour les URLs récupérées. Cette fonction génère un fichier `JSONLines` se trouvant dans le dossier */CrawlGooglePython/bin/searchEngine/output* et se nommant *unsortedOutput.jsonl*. Si l'on souhaite modifier les balises à étudier, il faut penser à modifier le fichier */CrawlGooglePython/bin/searchEngine/items.py*.

`getURLs.py` : c'est le script permettant de récupérer les n premières URLs affichées pour une requête donnée, il utilise le framework `Selenium`.

- **génération d'une fenêtre Firefox** : `Selenium` permet de simuler une requête Google en générant une fenêtre d'un navigateur donnée. Ici, on utilise Firefox.

- **génération de la requête** : une fois le navigateur lancé, la requête est créée à partir des mots-clés de l'utilisateur puis transformée en une URL. Ce lien spécifie également le nombre de résultats à afficher : si n est inférieur à 100 (limite imposée par Google quant au nombre de résultats à afficher sur une seule page de recherche), alors exactement n résultats seront présents sur la page générée. Sinon, une première page avec 100 résultats sera créée puis une seconde avec le nombre de résultats restants sera générée plus tard. La limite est donc de 200 résultats, mais on peut l'étendre en répétant simplement cette méthode.

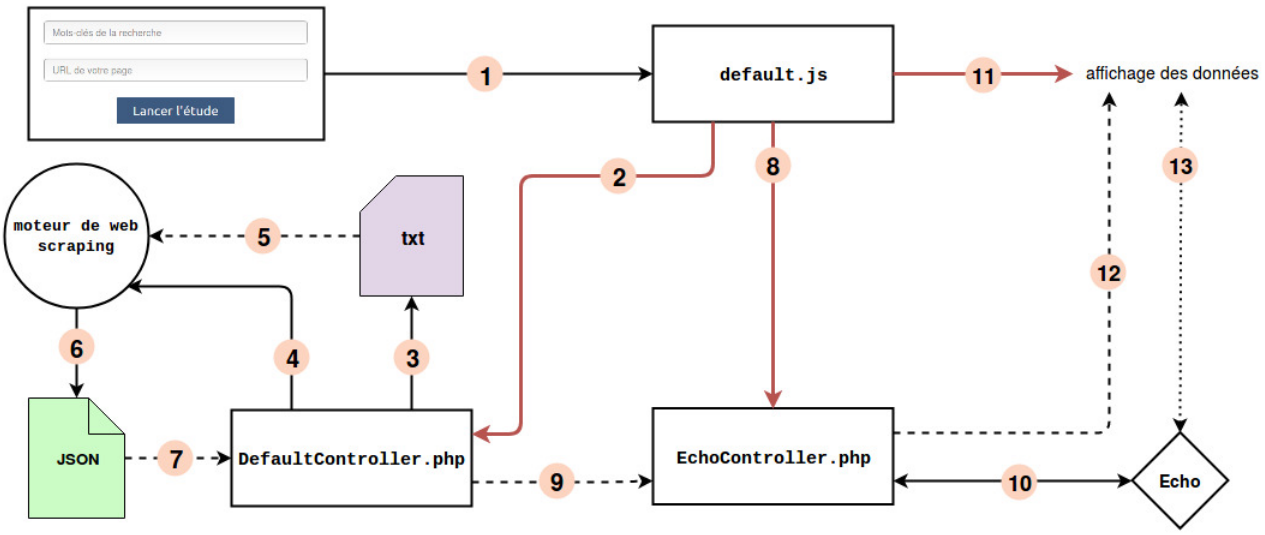
- **récupération des URLs** : une fois la page de recherche générée, le script permet de récupérer les URLs de chacun des liens présents en identifiant les balises qui identifient leur présence. Ces liens sont ensuite stockés dans une liste qui est renvoyée à la fin du processus.

- **gestion d'erreur** : entre temps, plusieurs erreurs sont gérées. On vérifie que la page n'est pas vide, que Google n'a pas supprimé de lien pour des droits d'auteurs ou encore qu'un lien vers Google image n'est pas présent. Dans le premier cas, le processus est arrêté. Pour les autres, un message d'information est affiché. Enfin, Google peut décider d'afficher un code captcha qui empêche la récupération des URLs et le script est capable de le repérer pour pouvoir informer l'utilisateur de ce problème.

3 L'apprentissage machine

pour plus d'information, se référer aux commentaires dans le code

Voici un schéma représentant l'architecture de cette première étape :



Description des étapes

- 1 : l'utilisateur spécifie ses mots-clés et l'URL de sa page
- 2 : une fois le bouton "Lancer l'étude" cliqué, l'appel aux requêtes Ajax est initié et la première requête Ajax exécute le contrôleur Symfony nommé *DefaultController.php*
- 3 : le contrôleur génère un fichier texte contenant les paramètres entrés par l'utilisateur en plus du nombre d'URLs et le degré de pertinence (200 et 30 par défaut)
- 4 : le contrôleur exécute le moteur de web scraping
- 5 : le moteur de web scraping enregistre les paramètres contenus dans le fichier texte précédemment créé
- 6 : le moteur de web scraping génère un fichier JSON contenant les données HTML des balises spécifiées, la position et l'indice de pertinence des URLs associées à la requête
- 7 : le fichier JSON est enregistré par le contrôleur sous forme d'un tableau PHP
- 8 : si le processus qui dure depuis l'étape 2 a fonctionné, alors une nouvelle requête Ajax exécute le contrôleur Symfony nommé *EchoController.php*
- 9 : le contrôleur *DefaultController.php* fournit au contrôleur *EchoController.php* le tableau contenant les données du web scraping
- 10 : une fois les données traitées, le contrôleur les envoie aux algorithmes Echo qui retourne les résultats de l'étude
- 11 : si le processus qui dire depuis l'étape 8 a fonctionné, alors une dernière requête Ajax permet de gérer l'affichage des données
- 12 : ces données sont renvoyées par le contrôleur *EchoController.php*
- 13 : au moment de l'affichage, il est possible d'exécuter à nouveau l'algorithme *EchoPos* afin de recalculer la position de la page avec les changements opérés

`default.js` : c'est le script qui fait le lien entre l'affichage des résultats et le calcul de ces derniers. Il va notamment permettre d'appeler les scripts `PHP` les uns après les autres une fois que l'utilisateur aura décidé de lancer l'étude grâce à des requêtes `Ajax` imbriquées. Cela permet à la fois d'optimiser la gestion d'erreur (puisqu'il est facile de savoir lors de quelle étape le processus a rencontré une erreur) mais aussi l'affichage (car les pages sont générées rapidement).

`defaultController.php` : c'est le premier script exécuté une fois le bouton de lancement cliqué. C'est aussi celui qui fait le lien entre l'étape de web scraping et celle de machine learning.

- **écriture du fichier de paramétrage** : le script permet de récupérer les données entrées par l'utilisateur avant de lancer le processus, soit son URL et ses mots-clés. Ces paramètres sont ensuite écrits dans le fichier texte que `scrape.py` va pouvoir lire plus tard. Ils sont accompagnés du nombre de résultats à étudier (*nResults*) et du degré de pertinence (*efficiency*). Ces valeurs sont codées dans le dur et peuvent être modifiées à tout moment.

- **lancement du web scraping** : une fois le fichier de paramétrage créé, le processus de web scraping présenté précédemment est lancé et les résultats sont récupérés depuis le fichier de sortie *output.json*.

`echoController.php` : c'est le script qui permet de formater les données puis de les envoyer à `Echo`.

- **conversion du fichier de sortie** : le fichier `JSON` (*output.json*) est récupéré puis transformé en un tableau `PHP` afin de pouvoir modifier les données qu'il contient. En réalité, il est même séparé en deux tableaux ; l'un contient les données de tous les résultats issus de la recherche Google (*arrayGoogleResults*), l'autre contient les données de la page de l'utilisateur (*arrayUserSite*).

- **récupération des meilleures pages** : le script est capable de récupérer les 10 pages les mieux référencées pour la requête donnée afin de pouvoir les afficher plus tard.

- **formatage des données** : le script permet de faire le lien avec `Echo`, mais ce dernier ne peut pas lire les données telles quelles. Un nouveau tableau (*tabResGoogle*) et donc créé afin d'y stocker les données modifiées de manière à ce que `Echo` puisse les lire. Ainsi, le tableau contient 4 champs pour chaque page : un identifiant (l'URL), une position, un degré de pertinence et une chaîne de prédicteurs. Cette chaîne est en fait composée de tous les termes confondus et avec comme suffixe le nom de la balise qui leur est associée individuellement (par exemple, *agence_TITLE*). Le même traitement est opéré pour la page de l'utilisateur.

- **duplication des pages pertinentes** : `Echo` se sert de la notion de pertinence pour pouvoir identifier les termes qui sont importants vis-à-vis du référencement. Un terme qui apparaît sur des pages pertinentes et non pertinentes n'est pas un terme important, mais un terme qui apparaît sur des pages pertinentes et non sur les pages non pertinentes l'est. Afin d'amplifier ce mécanisme, les pages pertinentes sont dupliquées de la manière suivante dans le code :

  - les pages allant de la position 1 à *n1* sont multipliées *coeff1* fois
  - les pages allant de la position *n1* à *n2* sont multipliées *coeff2* fois
  - les pages allant de la position *n2* à *n3* sont multipliées *coeff3* fois

  Toutes ces valeurs sont évidemment modifiables.

- **appel aux algorithmes** : une fois les données traitées, les algorithmes `EchoBT`, `EchoPos` et `EchoV` peuvent être appelés. En ce qui concerne `EchoBT`, il est possible de modifier le nombre de termes à renvoyer dans le fichier *echobt/Network.java* via la constante *nb*.

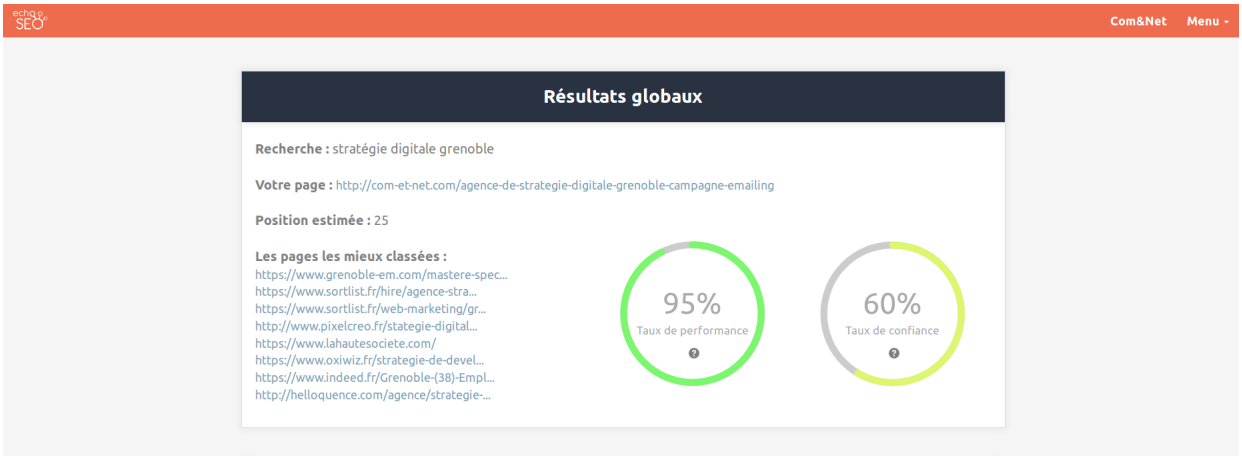
- **triPredicteurs** : les meilleurs termes renvoyés peuvent avoir des formes très variables. Dans certains cas par exemple, énormément de mots de la balise *alt* sont proposés. Ainsi, cette fonction permet de spécifier le nombre de termes à renvoyer par balise. Ces valeurs peuvent être modifiées librement à travers les constantes *nbTitle*, *nbUrl*, *nbH1*, ... De plus, cette fonction permet de renvoyer les résultats triése par ordre d'importance de balise (et par score au sein de chaque balise).

- **refreshAction** : cette fonction permet de recalculer la position de la page après que l'utilisateur ait modifié son contenu (voir plus loin). Le nouvel objet `JSON` contenant les nouvelles données est récupéré via une variable `POST`, puis le traitement des données est effectué pour ensuite appeler `EchoPos` et renvoyer le résultat.

4 L'UX design

pour plus d'information, se référer aux commentaires dans le code

Voici des captures de l'affichage des résultats :



Pour cette étape, la majorité du travail se trouve dans le fichier **default.js** situé dans le dossier */echoSEOBundle/Resources/public/js* (même si les fichiers `HTML` et `CSS` permettent d'y participer grandement). Une fois que toutes les requêtes `Ajax` se sont déroulées sans qu'aucune erreur ne se produise, la page de résultat est générée. Elle se divise en deux sections distinctes : la première comporte des informations générales vis-à-vis de l'étude menée, la seconde présente les meilleurs termes qui en ressortent.

Au niveau de la première section, deux graphes correspondant aux résultats renvoyés par `EchoV` sont générés. Leur couleur varie en fonction de la valeur du pourcentage qui leur est associé. Les 10 meilleures URLs sont également affichées ainsi que la position estimée de la page (`via EchoPos`) et les données entrées par l'utilisateur.

Quant à la seconde section, elle comporte les termes renvoyés par `EchoBT` triés par ordre d'importance de balise et par score au sein de celles-ci. Une couleur est aussi associée à ces termes et permet de rendre compte de l'importance de ces derniers. Chacune des sous-sections formées par les balises comporte un champ de texte dans lequel est écrit le contenu dans la balise correspondante de la page de l'utilisateur. Ce texte peut être modifié en se basant notamment sur les termes proposés. Une fois les changements opérés, l'utilisateur peut cliquer sur le bouton intitulé "Calculer la nouvelle position" afin d'estimer la position de la page si les modifications avaient vraiment eu lieues. Enfin, si l'utilisateur le souhaite, il peut les exporter via le bouton correspondant : un fichier `JSON` est alors proposé au téléchargement et il contient toutes les données inscrites dans les champs de texte (modifiés ou non). Il lui suffira ensuite de copier/coller ce contenu dans les balises associées (dans son code) pour pouvoir réellement améliorer le référencement de sa page web.