Heraldic Documentation

Version 0.0.1

Tagazoc

juin 20, 2019

Table des matières

CHAPITRE 1

Guide

1.1 Fonctionnement général

Le fonctionnement d'Heraldic est simple : les articles de divers sites Web sont récupérés via leur URL, via diverses méthodes (suivi de flux RSS, soumission manuelle, etc.). La page Web est alors analysée et divers contenus intéressants en sont extraits : titre, mots utilisés dans l'article, liens hypertexte, date de publication, etc. Ces données sont ensuite stockées via un moteur d'indexation pour permettre leur exploitation ultérieure parmi des centaines de milliers de données provenant d'autres articles.

1.2 Installation

Heraldic est développé en Python 3.6+. L'utilisation de **pipenv** est conseillée pour installer automatiquement les dépendances contenues dans le fichier *Pipfile.lock* :

```
git clone https://github.com/Tagazoc/Heraldic
cd heraldic
pipenv shell
pipenv update
```

1.3 Configuration

La configuration par défaut est présente dans le fichier *config/default.ini* :

```
[DEFAULT]
elasticsearch_host = 127.0.0.1  # Hôte elasticsearch
elasticsearch_port = 9200       # Port elasticsearch
extract_words = no              # Désactivation de l'extraction des mots des articles
```

La configuration locale s'effectue dans le fichier *config/config.ini* .

1.4 Récupération d'articles

La récupération d'articles par Heraldic s'effectue via un utilitaire en ligne de commande : :

```
python -m heraldic <commande> [options]
```

Actuellement, il existe deux commandes :

— « `gather` », qui permet la récupération d'un article, ou une liste d'articles dans un fichier. L'aide complète est :

```
usage: heraldic gather [-h] (-f FILE | -i | -u [URL [URL ...]]) [-d DEPTH]
                       [-o] [-t]

optional arguments:
  -h, --help            show this help message and exit
  -f FILE, --file FILE  File containing one or several URLs (one per line)
  -i, --stdin           Get URL from stdin
  -u [URL [URL ...]], --url [URL [URL ...]]
                        URL to gather
  -d DEPTH, --depth DEPTH
                        Depth of recursive gathering of sources
  -o, --override        Gather again up-to-date documents
  -t, --test            Stop on optional parsing exception
```

— « `harvest` », qui permet la récupération d'un flux RSS ou tous ceux qui sont enregistrés dans l'indexeur. L'aide complète est :

```
usage: heraldic harvest [-h] [-s] [-o] [-d DEPTH] [-i] [-t]
                    [-r RECURSIVE_STEP]
                    [media]

positional arguments:
  media                 Specify only one media to harvest

optional arguments:
  -h, --help            show this help message and exit
  -s, --sources         Gather the sources of indexed documents instead of
                        feeds
  -o, --override        Gather again up-to-date documents
  -d DEPTH, --depth DEPTH
                        Depth of recursive gathering of sources
  -i, --crawl-internally
                        Only crawl domains for this media
  -t, --delay           Time between document gathering (in seconds)
  -r RECURSIVE_STEP, --recursive-step RECURSIVE_STEP
                        Step between recursive crawling in gathered sources (0
                        disables)
```

On remarque l'option `DEPTH` qui permet la récupération récursive des liens dans les articles, si ceux-ci sont supportés bien entendu. L'option `sources` permet de récupérer, à la place d'un feed, les URL placés en source de documents déjà indexés. Dans ce contexte, l'option `RECURSIVE_STEP` détermine le nombre de documents à ainsi récupérer avant d'en effectuer une récupération récursive (celle-ci est plus efficace car elle inclut les sources du type « Lire aussi » ou « Pour aller plus loin » placées sur la page mais pas indexées).

— « `test` », qui permet la récupération simplifiée des articles spécifiés comme référence pour les différents extracteurs d'un ou de l'ensemble des médias supportés :

```
usage: heraldic test [-h] [media]

positional arguments:
  media       Specify only one media to test

optional arguments:
  -h, --help  show this help message and exit
```

1.5 Documents

Dans Heraldic, un document représente un contenu sur une page Web. Cela est généralement un article, mais cela peut aussi concerner un diaporama ou une page dédiée à une vidéo. Le processus de récupération d'un document est bien précis afin de minimiser la probabilité de doublon de documents dans l'indexeur.

1.5.1 Processus de récupération d'une URL

Vérification du support du domaine

La matière première pour la récupération d'un article est son URL, ou une autre URL qui redirige sur cette URL. La première vérification (outre l'intégrité de cette URL) est le support du domaine : il n'est pas question de permettre à quiconque de requêter n'importe quel site de la toile via Heraldic. Le domaine doit donc être utilisé par un média supporté par l'outil (par exemple : `www.lefigaro.fr`), ou un redirecteur accepté, qu'il soit spécifique (`lemde.fr` est le générateur d'URL courtes du Monde) ou plus générique (bit.ly est à envisager).

Il y a là un point important à noter ; il n'est pas envisageable, ni souhaitable, de pouvoir requêter n'importe quel article provenant de n'importe quel site Web, mais uniquement les sites sur lesquels un travail d'extraction a été effectué. Pourquoi ? Parce qu'il y a bien des manières de développer un site, ce qui rend de nombreuses données difficiles à récupérer sans une recherche humaine préalable ; et on ne saurait prendre l'intégralité des données présentes sur la page HTML que l'on récupère : ainsi, énormément de sites d'information insèrent des liens vers d'autres articles autour et à l'intérieur du corps même de l'article, pour provoquer la poursuite de la lecture vers un autre article. Il faut donc pouvoir identifier ces contenus qui ne doivent pas être considérés comme des sources. Par ailleurs, certaines données ne sont absolument pas normalisées et dépendent entièrement de l'équipe qui a créé le site web (par exemple, l'AFP doit être cité lorsqu'un article provient d'une dépêche, mais l'emplacement de cette donnée semble être totalement libre).

Ainsi le domaine est vérifié ; ensuite, on recherche la présence de cette URL dans l'indexeur. On s'affranchira du protocole utilisé (HTTP ou HTTPS), ainsi que des données « supplémentaires » que l'on peut trouver dans une URL : paramètres, port du serveur, balises. Si le document existe, on procèdera alors à sa mise à jour ; sinon, c'est un nouveau document qui sera stocké.

Redirections

Seulement à ce moment, l'URL (simplifiée) est collectée (avec la méthode HTTP « GET ») mais les vérifications ne sont pas terminées : il faut prévoir le cas où l'URL génère une ou plusieurs redirections, et vérifier l'URL de la cible. Une fois encore, on vérifie donc le domaine mais aussi la présence ou non de l'URL au sein de l'indexeur ; on vérifie aussi éventuellement la ressource de l'URL (`/guerre/les-guerres-sont-elles-justes.html`) pour voir si cela correspond bien à la façon de faire du site de référencer les articles, et si cela n'est pas une autre page sans contenu journalistique (par exemple un recueil comme `/dossier/guerre/`).

Extraction

Alors l'extracteur correspondant au site Web est identifié, puis commence le processus d'extraction ; chaque donnée considérée comme intéressante et référencée sur le site est récupérée. Comme chaque extracteur est créé par un être humain, il arrive fréquemment que l'extraction de certaines données se solde par une erreur. On distingue les données obligatoires qui empêchent complètement le stockage du document, et les données optionnelles qui le permettent tout de même. Néanmoins, ces erreurs sont stockées dans l'indexeur afin de pouvoir remédier ultérieurement au problème d'extraction et récupérer de nouveau le document.

Stockage

Les données sont alors stockées dans l'indexeur. Si l'URL existait déjà, on considère son évolution : si aucune donnée n'a évolué, rien n'est modifié. Si en revanche une ou plusieurs données ont été modifiées, le document existant dans l'indexeur est écrasé mais les anciennes données sont tout de même stockées dans l'indexeur à un autre endroit ; ainsi, pour un article mis à jour toute la journée et récupéré à chaque heure, il est possible d'avoir les données pour chacune des versions.

1.6 Extraction

1.6.1 Processus

Le processus d'extraction des données consiste simplement en la lecture du code HTML de la page contenant un article, et la récupération de données précises placées à divers endroits de ce code. Comme chaque site a une façon de présenter et de coder les données différente, ce processus est spécifique à chaque média ; le média concerné est déduit du domaine présent dans l'URL, après vérification de celle-ci comme valide et supportée. Un même média peut être déployé sur plusieurs domaines et surtout sous-domaines ; cependant, il peut y avoir plusieurs styles de pages au sein d'un même domaine. Un média dans Heraldic peut donc concerner plusieurs domaines et proposer plusieurs extracteurs ; un extracteur est composé d'une collection de fonctions pour extraire les données des articles lui correspondant.

Lorsqu'il y a plusieurs extracteurs, le choix est effectué en testant la présence d'un élément spécifique au style de page concerné par chaque extracteur. Une fois l'extracteur choisi (ou s'il n'y en a qu'un) l'extraction peut réellement commencer. Elle débute par l'extraction des attributs obligatoires, sans lesquels les données du document ne seront pas indexées, par exemple le corps de l'article (qui ne sera pas directement indexé, mais est nécessaire à la récupération de nombreux autres attributs), et se poursuit par l'extraction des autres attributs. Chaque erreur d'extraction est enregistrée afin de permettre sa correction ultérieure.

1.6.2 Attributs à extraire

L'ajout d'un média dans Heraldic s'effectue par l'analyse des différents styles de pages présents sur le ou les sites Web de ce média. Il faut faire une copie du fichier canevas heraldic/media/pattern.py et renseigner les informations suivantes :

Classe « *Media* » : Elle comprend les informations de base qui concernent le média.

- `supported_domains` : la liste des domaines liés à ce média.
- `id` : l'identifiant utilisé en base. Il s'agit généralement du nom du média, en minuscule, avec des tirets bas (ou underscore : _) pour remplacer les espaces ou ponctuations.
- `display_name` : le nom du média.
- `articles_regex` : une liste d'expressions régulières permettant d'identifier au mieux l'URL d'un article (en opposition avec d'autres pages du site qui n'en sont pas). Cela peut concerner une extension, un identifiant numérique, etc.

Classe « *Extracteur* » : Comme son nom l'indique, elle contient les méthodes permettant l'extraction. Elle est spécifique à un style de page et il peut y en avoir plusieurs. Lorsque c'est le cas, il faut implémenter la méthode _check_extraction qui doit renvoyer (comme booléen) la présence ou non d'un élément du code HTML spécifique au style de page concerné.

Les autres méthodes à implémenter correspondent aux attributs extractibles du document, avec pour nom « `_extract_<nom de l'attribut>` ». Certains attributs sont généralement implémentés de la même manière (dans la classe parente GenericExtractor) et n'ont pas besoin d'être réimplémentés, les autres doivent l'être. Les attributs sont :

- `body` (obligatoire) : Le corps effectif de l'article, sans le titre, éventuellement avec les légendes des images. Les contenus ajoutés par le site (liens d'actualité, publicités) doivent être supprimés, sauf s'il s'agit bien sûr de sources directement liées à l'article. Un traitement supplémentaire supprime automatiquement certaines balises internes telles que les balises « script ». Cet attribut n'est pas directement indexé, mais il est conservé dans l'attribut de la classe « _body_tag » qui peut être accédé dans d'autres méthodes d'extraction.
- `title` : Le titre de l'article. L'intégration des articles avec les réseaux sociaux et les sites d'agrégation de contenus de presse fait que cette donnée est toujours implémentée de la même manière, sauf exception. On tentera de ne pas retenir le nom du média, qui peut être inséré dans le titre.
- `description` : La description de l'article : il s'agit généralement d'une méta-donnée qui n'est pas systématiquement affichée sur la page. Comme pour le titre, il y a rarement besoin de réimplémenter son extraction.
- `category` : La catégorie de l'article selon le site Web. Cela peut ainsi être aussi bien « Actualité » que « Monde » ou « Cuisine ». On trouve généralement cette donnée dans une barre de navigation (souvent appelée « navbar »).
- `doc_publication_time` : La date et l'heure de publication de l'article (à ne pas confondre avec gather_time, la date de récupération par Heraldic). On pourrait penser que cette donnée est aussi standardisée que le titre ou la description, mais ce n'est pas le cas : il y a ainsi quatre façons différentes de l'extraire qui sont implémentées dans la classe parente. Cela couvre néanmoins la plupart des sites d'information.
- `doc_update_time` : La date et l'heure de mise à jour de l'article sur le site (à ne pas confondre avec update_time, la date de mise à jour dans Heraldic). Elle est optionnelle et suit généralement la façon de faire de la date de publication.
- `keywords` : Les mots-clés associés par le site à l'article. Ils peuvent être nombreux, car ils interviendront dans le référencement sur les moteurs de recherche. C'est une méta-donnée qui ne nécessite généralement pas de réimplémentation.
- `href_sources` : Les liens placés en source de l'article, généralement situés directement dans le corps de l'article. Il convient de bien distinguer les sources de l'article des liens qui ne sont que des références vers d'autres endroits du site (par exemple un lien sur le nom d'une personnalité, pointant vers une page contenant une liste d'articles sur cette personnalité), et surtout des liens ajoutés du type « Lisez aussi ». Ces derniers peuvent être conservés dans l'attribut de classe « _side_links » en vue de récupérer d'autres articles qui pourraient être absents d'Heraldic. Un traitement supplémentaire simplifie l'extraction : il faut juste récupérer les balises « a » caractérisant les liens dans la fonction d'extraction.
- `news_agency` : L'agence de presse à l'origine de la dépêche qui donne lieu à l'article, si elle existe bien évidemment. Généralement AFP, ou Reuters. Chaque média écrit cette information à l'endroit qu'il souhaite (avec le nom de l'auteur par exemple) mais toujours au même endroit.
- `subscribers_only` : Si oui ou non l'article est réservé aux abonnés ; dans ce cas, les quelques lignes qui sont généralement proposées gratuitement pourront être identifiées comme n'étant pas l'article complet.
- `document_type` : le type de document, pour le moment « article », « video » ou « panorama ».
- `side_links` : Les liens dans un article vers d'autres articles du même média, qui ne sont pas des sources de cet article. Leur extraction est généralement effectuée avec celle de l'attribut href_sources.

1.7 Indexation

1.7.1 Concepts

Une fois qu'un article a été récupéré et que ses attributs ont été extraits de son contenu, ces derniers sont stockés dans une solution d'indexation, à savoir Elasticsearch. C'est un produit open source, qui dispose de fonctionnalités payantes comme le contrôle d'accès. Ce projet étant conduit avec quasiment aucune dépense financière, il n'est pas envisageable (en termes de fonctionnalité payantes, donc, mais aussi de puissance de serveur) de permettre à tout ceux qui le souhaitent d'accéder à notre instance Elasticsearch ; c'est notamment pour cette raison qu'il y a une API spécifique à Heraldic.

L'idée générale est donc d'encourager chacun souhaitant jouer comme il le souhaite avec toutes les données (plus que via l'API) à déployer une machine virtuelle avec une instance Elasticsearch. Pour comparaison, notre instance actuelle d'Elasticsearch tourne sur une distribution CentOS 7 hébergée sur un VPS avec deux coeurs, 4 Go de mémoire vive et 20 Go de mémoire SSD.

Tout le contenu indexé sera disponible en ligne et pourra être synchronisé avec un script, sur un modèle d'une sauvegarde complète chaque semaine et une sauvegarde incrémentale chaque jour. Ainsi, il sera très simple d'obtenir un environnement rigoureusement identique à celui de référence. En outre, cela permettra à quiconque souhaite utiliser la solution pour ses propres données et articles, de le faire sans rendre pour autant ces données publiques.

1.7.2 Indexes

On trouve différents index correspondant à différents types d'objets à indexer :

- `docs` : L'index principal, qui contient les données correspondant aux attributs des documents récupérés.
- `docs_history` : L'index contenant les anciennes versions des attributs des documents mis à jour. Par exemple, si le titre d'un document est mis à jour, sa récupération par Heraldic modifiera le document dans l'index docs et créera un document dans l'index docs_history contenant uniquement le titre avant modification (ainsi que la date de la précédente version).
- `errors` : L'index contenant les erreurs générées durant l'extraction d'attributs suite à la récupération d'une URL. On y trouve des erreurs liées à des documents existants dans l'index docs, mais aussi des erreurs sur des documents qui n'ont pas du tout été indexés à cause de l'échec de l'extraction d'un attribut obligatoire.
- `feeds` : L'index contenant les feeds utilisés pour la récupération des articles via des flux RSS. On y conserve la date et l'heure du dernier accès au flux, afin de ne pas parcourir l'ensemble du flux RSS s'il n'a pas été mis à jour depuis.

La création des indexes n'est en soi pas obligatoire pour l'indexation des documents, néanmoins elle est requise pour spécifier les types des propriétés des objets indexés (mapping). Pour ce faire, il suffit de lancer le script « `heraldic/scripts/create_indices.py` ».

La création des feeds se fait via un autre script, « `heraldic/scripts/create_feeds.py` », en utilisant un fichier CSV contenant la paire « identifiant du média ;URL du feed » pour chaque feed.

1.8 API

Une API était nécessaire pour accéder aux données principales contenues dans l'indexeur, tout en restreignant la possibilité de faire des requêtes longues et complexes, sans parler des suppressions de contenu qui ne sont pas limitables avec la version open source de l'indexeur, lorsqu'on y a un accès direct.

Cette API est construite avec le framework Connexion et le standard de description OpenAPI. Elle se lance via le script `heraldic/api/server.py` .

Pour y accéder, il suffit d'accéder à l'adresse http://localhost:5000/api (le port est spécifié dans le script `server.py`). La documentation complète de l'API est disponible sur l'UI Swagger disponible à l'adresse http://localhost:5000/api/ui , ainsi que des fonctionnalités de tests de chaque requête.

CHAPITRE 2

Indexes et tables

— genindex
— modindex
— search