Présentation du travail effectué par le groupe 2 : Identification des nouvelles sources pertinentes

Ce document a pour objectif de coordonner les différentes parties et codes fournis par ce groupe.

Ce projet se place dans le cadre du projet inter-promo 2021 de la formation SID de l'université Paul Sabatier, Toulouse. En collaboration avec Berger-Levrault, le projet Helios a pour but de proposer une solution automatique à de la veille technologique autour de l'innovation dans la gamme de gestion.
Le groupe 2 "Identification des nouvelles sources pertinentes" a pour but de mettre en œuvre une solution de détection de nouvelles informations automatiquement. Cette ligne vise à soulager l'humain de chercher continuellement de nouvelles informations concernant son sujet thématique.
Cette problématique d'automatisation soulève plusieurs autres problématiques :

- Comment chercher de manière automatique les nouveaux sites ?
- Comment définir la pertinence d'un article ou d'un site ?
- Comment rendre de manière automatique le traitement des nouvelles sources ?
- Comment récolter de manière universelle tous les sites visés ? (scrapeur générique)

Le résultat de ce projet vise à répondre à l'ensemble de ces questions en apportant une ébauche de réponse. Une partie de ces réponses ont pu être mise en chaîne d'automatisation tandis que les autres ne sont qu'à l'état d'ébauche, de preuves de concept.

Identification des nouvelles sources pertinentes

Cette partie s'articule autour des deux premières problématiques.

Comment chercher de manière automatique les nouveaux sites?

La solution qui a été mise en place pour cela est d'utiliser la puissance des moteurs de recherche en donnant une équation de recherche textuelle et nous renvoyant une liste d'articles pertinents.
Pour cela nous avons visé le moteur de recherche Google jugé mondialement comme le moteur le plus efficace existant actuellement. Sa robustesse et pertinence des réponses nous a semblé être le meilleur choix.
Néanmoins, pour cela, nous avons besoin d'une API pour ne pas être bloqué par les contrôleurs de Google. En effet, Google bloque facilement les individus qu'il considère comme des robots (ce qui est notre cas). Nous avons opté pour l'API nommé scraperAPI

[voir https://www.scraperapi.com/] qui propose une version gratuite permettant de faire 5000 requêtes par mois ce qui nous semblait suffisant pour le projet.

La première étape est donc de créer un ensemble de requêtes google. Nous nous basons donc sur le produit cartésien des lexiques d'innovation et de gamme de gestion afin de couvrir l'ensemble des thèmes. Ces équations de recherches essayent de prendre en compte les formes fléchis des mots [voir G2_serveur_v4/Identification_Des_Nouvelles_sources/g2_Create_Word_Combination_v6.py].

Une fois les équations construites, il revient donc de faire les recherches googles. Pour cela, les paramètres d'entrée sont :

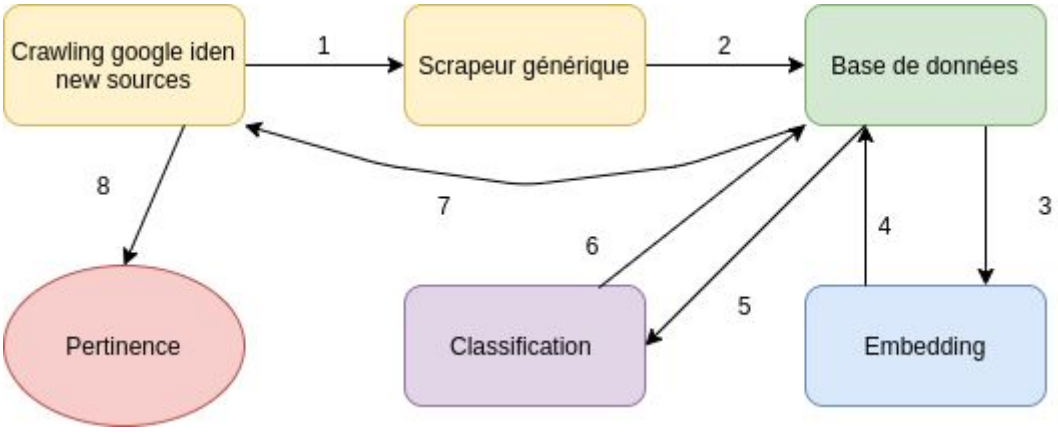
- la clef API [voir G2_serveur_v4/Identification_des_nouvelles_sources/API_key.txt]
- les paramètres d'entrée qui définissent le nombre de requêtes, le nombre de couple par requête et le nombre de résultats maximum souhaités [voir G2_serveur_v4/Identification_des_nouvelles_sources/Parameters.py]
- la date du dernier crawling [voir G2_serveur_v4/Identification_des_nouvelles_sources/date_last_crawling.txt]

Ces données sont automatiquement récupérées par le lanceur du crawling définis en fonctions dans le fichiers G2_serveur_v4/Identification_des_nouvelles_sources/g2_Launch_Crawler_v6.py.

Comment définir la pertinence d'un article ou d'un site ?

Cette réponse ne fut pas évidente à trouver. Plusieurs solutions ont été proposées et peuvent se combiner. Le but est de donner un score de pertinence aux articles et de les agréger (avec une somme ou une moyenne) par site afin de définir la pertinence du site.
Le document /POC/POC_mesures_pertinence_v3.ipynb résume les différentes mesures que nous avons pu trouver.

La dernière mesure proposée dans ce POC est d'utiliser les résultats de classification (faites par le groupe 5 du projet) donnant les probabilités qu'un document parle d'innovation ou de



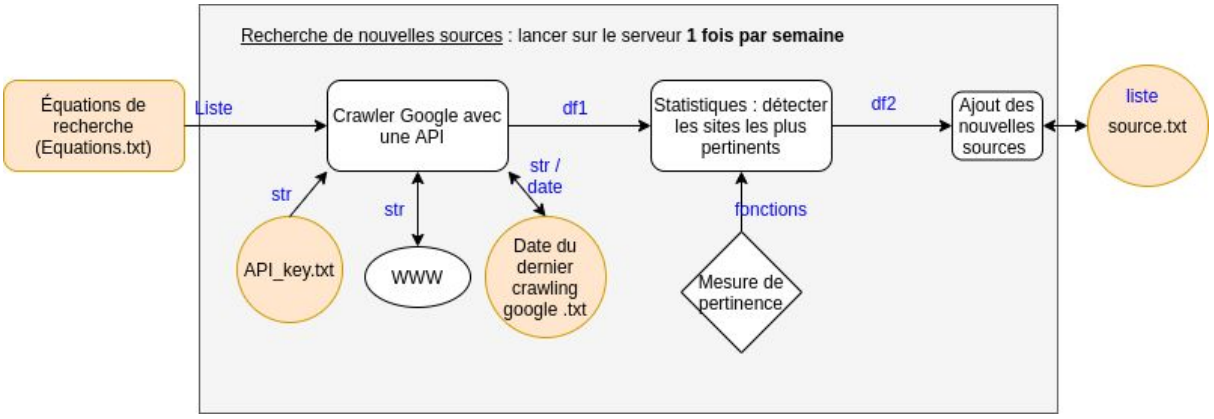
gestion. Des mesures sont proposées sur le POC. La figure 1 présente l'architecture globale

et du Groupe 2 qui répond à cela. Les données seraient transmises aux différentes modélisations et peuvent être retournées à l'identification de la pertinence grâce à un attribut spécifique placé dans la BD.

Une partie des mesures ont été aussi mises en production dans le fichier G2_serveur_v4/Identification_des_nouvelles_sources/g2_Launch_Pertinence_v5.py.

Automatisation des processus d'identification des mesures pertinentes

Une automatisation a été mise en place pour ce processus. Pour cela il suffit d'exécuter le fichier G2_serveur_v4/G2_pipeline_nouvelles_sources.py. La commande Cron suivante permettra de faire tourner ce processus une fois par semaine. Nous avons choisi arbitrairement le lundi à minuit :
``` 0 0 * * 1 python3 /G2_serveur_v4/G2_pipeline_nouvelles_sources.py ```

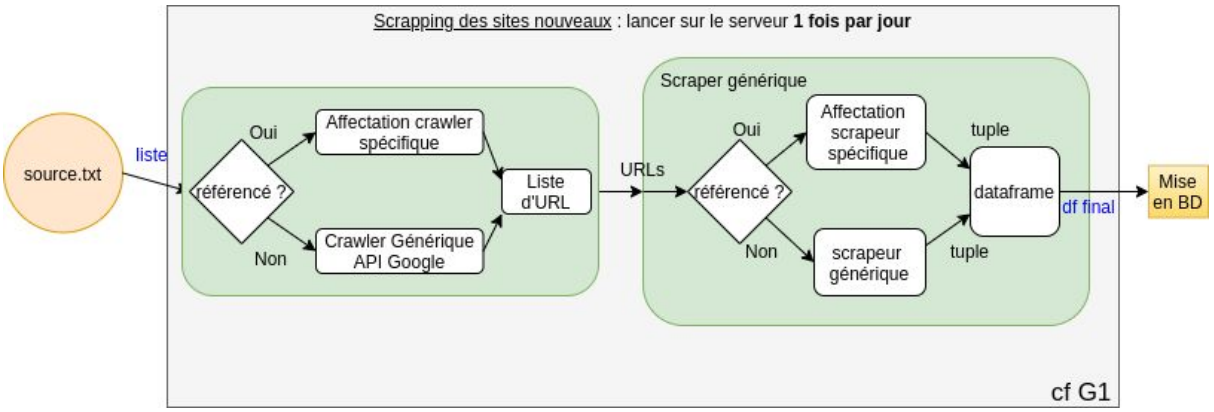


Mettre en place un scraper automatique et ajustable

La difficulté dans ce type de projet automatique est de faire un récolteur de données automatique. Les architectures de chaque site sont suffisamment différentes pour que les robots actuels de scrapping ne puissent pas fonctionner d'un site à l'autre.
Or dans un projet comme celui là où la liste des sites à scraper a vocation à augmenter, il est nécessaire de mettre en place un crawler et scraper générique que l'on pourrait appliquer à chaque site.

Crawler générique par site

Nous avons dans un premier temps fait un crawler pseudo générique puisqu'il n'est le fruit que de la normalisation d'un code de crawling et applicable grâce à une base de
```

connaissance (voir G2_serveur_v4/Données/tags_crawl.json) à un nombre de site référencé (dans la base de connaissance). Cette version a été mise en production et se trouve dans le fichier G2_serveur_v4/Scrapping_nouvelles_sources/g2_crawl_by_site_v1.py. Cette solution nécessite l'intervention humaine afin d'ajouter les balises et informations dans la base de connaissance.
Une optimisation par filtrage temporel en amont du scraping a été abordée dans la POC POC/g2_poc_date_filtration_optimisation_v1.ipynb.

Cette solution proposée n'est pas générative. Le fichier POC/G2_Crawler_API_Google_Quotidien_POC.ipynb propose une solution générique qui pourra s'appliquer sur tous les types de sites et avoir un filtrage par date en amont du scrapping.

Scraper générique : l'objectif ultime

Avoir un scraper générique permet de trouver une solution à la plupart des problématiques engendrées dans ce sujet. Il permet d'assurer la pertinence d'un article mais aussi de rendre le processus parfaitement automatique.

La première version du scraper générique est pseudo-générique car elle nécessite l'appel de l'humain pour faire un ensemble de scraper spécifique aux sites visés. L'architecture de ce 'Big Scraper' est de faire un appariement entre les sites et le scraper spécifique qui lui est dédié. Il n'est pas possible de l'utiliser avec un nouveau site non référencé.

La deuxième version reprend en partie ce Big Scraper mais lorsque le site n'est pas référencé il passe dans le scraper générique qui essaye de trouver les informations. Ce dernier n'est pas très précis et peut ne pas être performant sur certains sites.
Cette solution a été mise en production (voir le fichier G2_serveur_v4/Scrapping_nouvelles_sources/BigScraper.py).
Une version plus abouti et un comparatif de plusieurs méthodes ont été dévelopé sur le POC dédié au scraper générique.

Automatisation des processus

Une automatisation a été mise en place. Pour cela il suffit d'exécuter le fichier suivant G2_serveur_v4/G2_pipeline_scraping.py)
Nous avons envisagé de faire tourner ce processus une fois par jour, fréquence à laquelle il y a assez d'articles à récolter. Pour cela, il suffit de faire la commande CRON suivante :
``` 0 12 * * * python3 /G2_serveur_v4/G2_pipeline_scraping.py ```
Les tests à plus grande échelle nous donnent le résultat suivant (sur un ordinateur classique avec une connexion internet limitée) : 100 requêtes ont été effectuées sur 17 sites en 3h et ont retourné 9285 par exemple.
```



Autres améliorations envisagées dans le projet

Dans un contexte où lancer une requête google est chronophage, il est pertinent de réduire le nombre de mots clefs utilisés. Mais si un mot clef n'est pas pertinent aujourd'hui, quelle garantie avons-nous qu'il ne le soit pas demain ? C'est à cette question que nous avons essayé de répondre dans la POC suivant POC/g2_crawling_POC_Popularite_Couples.ipynb

Une autre préoccupation est de se questionner sur le moteur de recherche. La POC POC/g2_launch_crawler_scholar_POC_v0.ipynb adapte le crawler google sur le moteur de recherche Google scholar.

Une partie des améliorations n'ont pas été initiées:

- Définir un moyen de ne plus rendre pertinent un site. Il faut donc réévaluer sa pertinence périodiquement afin de ne pas surcharger la liste des sites pertinents inutilement.
- Réduction du temps d'exécution. Essayer de réduire le temps d'attente entre les requêtes dans l'identification des nouvelles sources pertinentes.
- Optimiser les explorations des sites pertinents. Si un site ne publie pas d'article tous les jours, il ne sera pas nécessaire de le consulter quotidiennement mais sur des périodes plus grandes. Cette décision doit pouvoir être adaptable à la temporalité d'un site. S'il se met à publier beaucoup d'articles soudainement, cette méthode devra donner une fréquence de récolte plus importante. Adapter la fréquence des récoltes de données.
- …