17 August 2019
Alexandre ALLANI

Encadrant : Jean PRUVOST, consultant en datascience
Conseiller d'étude : Romain BILLOT
Responsable filière : Cécile BOTHOREL

Rapport de stage de fin d'étude

Consultant en datascience chez Sia Partners

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

1 Résumé

2 Summary

Contents

3 Introduction

4 Présentation de l’entreprise

5 Mission : webscrapping pour Cleep (1 mois)

Cette partie est consacrée à la première mission que j'ai effectué sur le sujet du webscrapping pour Cleep. Le webscrapping est la collecte de données/informations sur des sites internet via un ou plusieurs scripts. Le but sera alors principalement de "piocher" dans le code source du site internet à webscrapper, ou via les requêtes effetuées par le serveur, les données que l'on souhaite récupérer.

Cette mission mêle à la fois des problématiques de récupérations de données, mais aussi d'automatisation de webscrapping via certains algorithmes que j'ai développé.

5.1 Présentation de Cleep

Cleep est une start-up créée en 2017, développant une application éponyme sur le sujet de l'achat en ligne. L'application permet de sauvegarder un produit repéré sur un site marchand quelconque (que ce soit un site généraliste comme Amazon ou un site spécialisé comme Asos). Cette application mobile est disponible sur mobile (iOS & Android) mais aussi via un site Internet [1]. Le principe de l'application est de pouvoir enregistrer n'importe quel produit pour ensuite le retrouver dans l'application. Il est alors possible de consulter via cette dernière le prix, la description, le nom, le site d'origine ainsi que les images reliés à ce produit. Le but de l'application est donc de faciliter l'achat de produits en ligne en ne passant que par une seule et unique plateforme. Par ailleurs, il est possible de créer une liste de "Cleep" (ie de produit) et de partager ces listes. Les listes publiques peuvent être trouvées via le moteur de recherche intégré.

Figure 1: Logo de Cleep

Sia Partners a investi 400000€ dans Cleep fin 2018, et travaille en collaboration sur différents sujets, notamment en matière de datascience. En plus du côté simplification du shopping en ligne, Cleep avec ses listes et son moteur de recherche permet de suivre des listes plus ou moins influentes. Ainsi l'application permet de se tenir au courant des modes actuelles et de celles à venir. Cela explique l'investissement de Sia Partners.
Vocabulaire spécifique à Cleep :

- cleep : Produit en ligne, enregistré dans la base de donnée. Un cleep est composé d'un prix, des images associés au produit, d'une description, du nom du produit ainsi de du site internet d'origine.
- liste de Cleep : Liste de Cleep, pouvant être partagé entre plusieurs utilisateurs. Chaque cleep est enregistré dans une liste.
- cleeper : Action d'enregistrer un produit dans l'application

J'ai travaillé pendant près de 3 mois pour Cleep. Je communiquais principalement avec Damien Meurisse côté Cleep et Paul Saffer côté Sia Partners.J'ai travaillé sur deux sujets :

- Webscrapping : Acquisition et traitement des données. Cela est fondamental pour le fonctionnement de Cleep
- Moteur de recommandation. Avant mon arrivé, aucun moteur de recommandation n'existait. Cette fonctionnalité était de plus en plus nécessaire afin que l'utilisateur puisse chercher de nouveau cleeps autrement que par le moteur de recherche présent dans l'application.

5.2 Objectifs et problématique

Cleep est une application reposant sur les articles et produits qu'elle permet d'enregistrer, mais aussi sur les méta-données qui leurs sont reliés. En effet, en allant sur Cleep, l'utilisateur doit pouvoir retrouver l'ensemble des informations qu'il recherche, sans pour autant retourner sur le lien de l'article. Il existe deux moyens d'enregistrer un produit :

- Par capture d'écran du site : l'image est enregistré dans l'application. L'utilisateur peut alors renseigner lui même les informations dont il a besoin (ie le prix, le nom de l'article, etc.). Le cleep créé est composé de la capture d'écran et des informations renseignés.
- Avec le lien du site : Grâce à un procédé, détaillé plus tard, le produit avec ses images et ses méta-données sont cleepés.

En général l'enregistrement via capture d'écran n'est pas optimal pour l'application. En effet, les utilisateurs prennent rarement le temps d'enregistrer toutes les informations, ce qui les intéresse en priorité est de garder l'image et le nom du produit dans l'application afin d'avoir un souvenir de ce dernier. Le cleep ainsi enregistré n'est pas utile pour Cleep car manquant de beaucoup d'informations.
En effet, Cleep possède une base de donnée dans laquelle les produits sont enregistrés avec leur méta-données. Un produit est composé des éléments suivant :

- URL du site internet
- Nom
- Prix d'origine
- Prix avec réduction (si disponible)
- Description (si disponible)
- URL des différentes images

Un cleep et un produit sont deux entités différentes. Un produit est une entité composée des éléments énoncés précédemment, tandis qu'un cleep est une entité pouvant faire référence à un produit créé par un utilisateur et avec lequel l'utilisateur interagit. Par ailleurs le produit est définit de manière unique par son URL. Par exemple, les mêmes chaussures disponibles sur un site A et sur un site B sont considérés comme deux produits différents. Le schéma ci-dessous explique la structure.

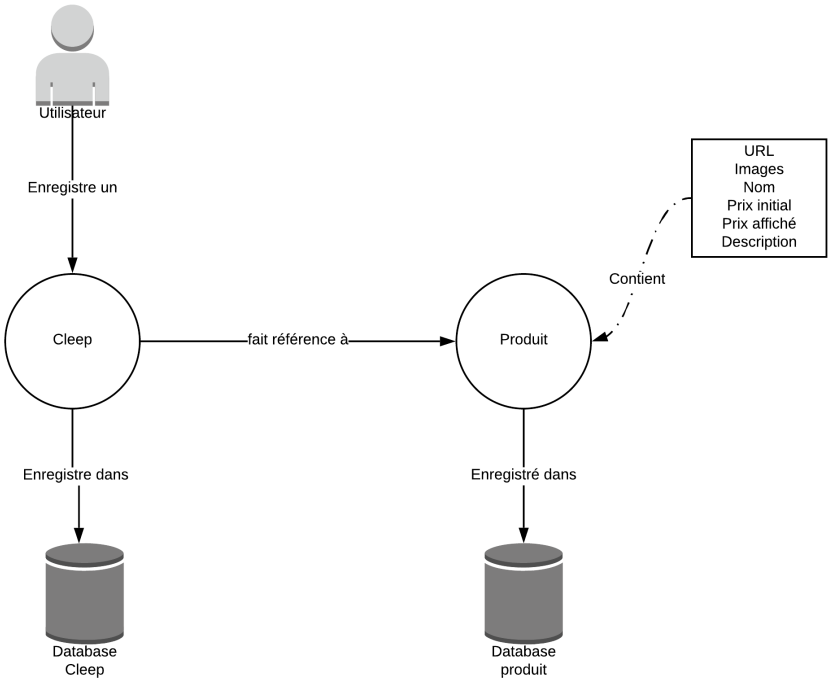

Figure 2: Organisation des données de cleep

Relier un cleep à un produit est un défi pour l'application, car les utilisateurs sont plus enclins à rester dans l'environnement Cleep lorsqu'ils ont les informations à leur disposition dans l'application. En effet les cleeps contenant ces information, l'utilisateur est moins tenté de suivre le lien du produit et a plus de chance de rester sur l'application et parcourir les listes de clepp d'autres utilisateurs.
Cependant pour pouvoir faire cela, il faut récupérer les informations des produits au préalables. C'est ici qu'intervient le webscrapping. Les méta-données qui nous intéresse sont récupérées par ce moyen et entré dans la base de donnée. L'objectif de cette mission était donc d'enrichir la base de donnée des produits de manière automatique.

5.3 Travail effectué

Cette sous-section relate le travail effectué sur cette mission.

5.3.1 Contexte

Avant mon arrivé, plusieurs methode de webscrapping étaient déjà utilisées:

- Méthode automatique
- Méthode spécifique

Méthode automatique:
Cette méthode était la plus utilisée pour récupérer les informations des produits. L'utilisateur rentre un lien dans l'application, par la suite ce lien est envoyé au serveur sur lequel la méthode est implémentée. Le serveur génère un navigateur internet sans interface graphique (headless browser) et requête le lien internet rentré par l'utilisateur. Ensuite, en analysant les balises du code HTML chargé, l'algorithme par certaines règles essaie de récupérer les méta-données. Ces informations sont alors transférés à l'utilisateur. Le schéma ci-après décrit le processus. Bien que cette méthode de scrapping soit indépendant du site internet, elle reste néan-

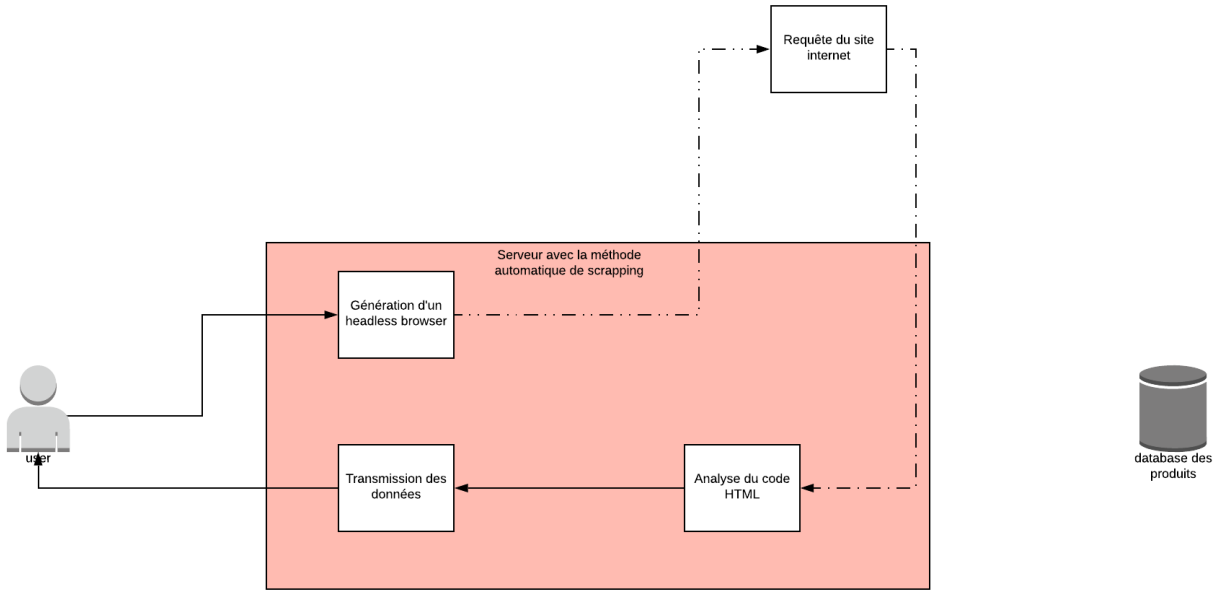

Figure 3: Méthode automatique

moins assez "précaire", car il est rare d'avoir toutes les informations nécessaires à disposition. Il est fréquent d'avoir certains attributs manquant, car les règles créées pour l'analyse du code HTML n'était pas assez vaste. Prenons les deux sites internet suivant :

Site N°1:

```
<!DOCTYPE html>
<html>
<head>
    <meta name="price" content="99EUR">
</head>
<body>
    <h1>My shopping website</h1>
    ...
</body>
</html>
```

Site N°2:

```
<!DOCTYPE html>
<html>
<body>
    <h1>My shopping website</h1>
    <p>Prix: 99 EUR</p>
</body>
</html>
```

Dans le cas N°1, il est facile de mettre une règle pour récupérer le prix dans le meta tag. Dans le cas N°2 cependant, le prix est "juste" inclut dans une balise <p>, rendant la règle précédente inutile. La méthode automatique implique donc beaucoup d'erreurs et d'approximations. Pour des soucis de mémoire, ces données ne sont pas enregistrées dans la base de donnée des produits, afin de ne pas garder les "fausse" données récupérées. Cette méthode pose un autre soucis, c'est son temps de réponse. En effet, afin de scrapper le site, le serveur est obligé de générer un headless browser, de faire la requête, analyser le code HTML pour finalement retourner la réponse. Le temps de réponse doit être relativement bas, car l'utilisateur au moment du cleep doit pouvoir continuer ses activités sur Internet sans être interrompu trop longtemps par le temps que met Cleep à cleeper son produit, et donc eviter toute frustration.

Méthode spcéifique:

C'est sur ce sujet que s'est concentré mon travail. Le principe de cette méthode est de prendre les sites internet un à un et d'appliquer un script de webscrapping qui leur est propre. Cela permet d'éviter les problèmes rencontrés précédemment. La structure du code HTML derrière les pages produits est dans la quasi-totalité des cas la même pour un nom de domaine particulier. Il est possible que selon les sous-domaines d'un site cette page varie, par exemple pour la fnac on a:

- https://livre.fnac.com/a13195869/Agnes-Martin-Lugand-A-la-lumiere-du-petit-matin
- https://jeux-video.fnac.com/a13608859/LEGEND-OF-ZELDA-LINKS-AWAKENING-FR-SWITCH-Jeu-Nintendo-Switch
- ...

Dans ce cas, on distingue les différents cas et ensuite on analyse le code HTML de chacune de ces pages. Cette méthode est bien plus fiable et donne de meilleurs résultats sur les méta-données requise.
Par ailleurs, afin de réduire le temps de réponse, une difficulté est rajouté : la génération d'un headless browser est supprimée m'obligeant à travailler avec le code source de la page (ie le résultat d'une requête cURL à la page), et les requêtes API.
En effet la génération d'un navigateur permettait de faciliter la tâche, car le code HTML étant totalement généré, il suffisait de récupérer le contenu des balises afin d'obtenir les informations nécessaires. Cependant via une requête cURL (ie le code source de la page), les script JavaScript ne sont pas déclenchés, il faut donc chercher au travers des differents script et requêtes API les éléments que l'on souhaite récupérer.

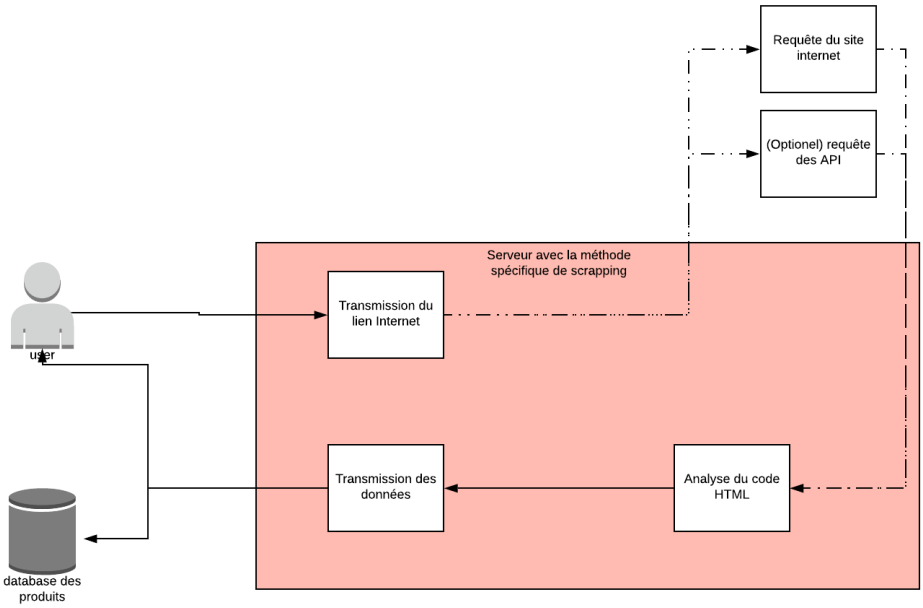

Figure 4: Méthode spécifique

Démarche

Le processus Figure 4 montre les différentes étapes de la méthode spécifique. Mon travail s'est principalement concentré sur l'algorithme analysant un site web en particulier et la gestion des requêtes.
Prenons l'exemple suivant [2] : Les encadrés rouges correspondent aux éléments que l'on doit récupérer. Vous

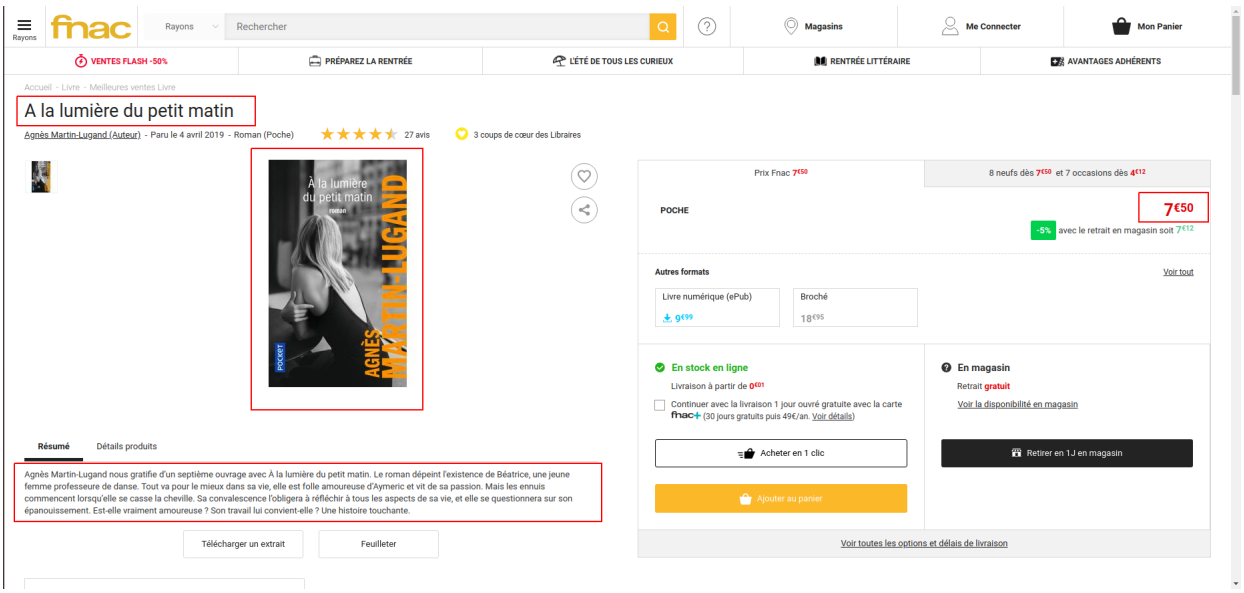

Figure 5: Exemple fnac

pouvez retrouver en annexe une version allégé du code HTML (Annexe 9.1). Il est alors facile de comprendre que récupérer l'attribut $data-src-zoom$ de la balise <li> nous permet de récupérer l'image et le JSON quant à lui permet de récupérer les autres éléments requis.
La démarche adopter pour scrapper les différents sites à toujours été la même. Tout d'abord on analyse le code source afin de savoir quels éléments sont à récupérer. Ensuite on teste avec plusieurs liens différents afin de tester la robustesse de l'algorithme. Une validation finale était faite côté client, et ensuite l'algorithme était mis en production.

Le dernier point nécessitant du travail : où trouver les liens permettant de scrapper les produits et peupler

la base de donnée? L'ensemble des liens déjà cleepés sont enregistrés, et donc peuvent être utilisé pour répondre à cette question. Le client possédait déjà un script permettant de trier les liens par nom de domaines. Ainsi pour chaque nom de domaine que j'ai traité, j'ai du trié les liens pertinents (ie de produits) des liens inutiles (page principal, page de catégories etc). Par exemple, pour la fnac certains utilisateurs ont cleepé des pages présentant plusieurs produit [3] ce qui ne convient pas au processus détaillé auparavant.

Résultats

A la fin de cette mission, j'ai développé un script de webscrapping pour plus de 56 sites différents. Cela a permis à Cleep de peupler sa base d'environ 220 000 produits différents.
Sur une vision plus large, les utilisateurs passent en moyenne plus de temps sur l'application (effet recherché), mais aussi ils ont tendance à cleeper plus de produit sur les sites dont le webscrapping a été fait.

5.4 Vision critique et apport personnel

La mission de webscrapping était plutôt simple, dans le sens qu'une fois les notions de webscrapping acquises, cette mission consistait à analyser un code HTML et les requêtes faits en javascript, pour trouver les informations nécessaires. Mon apport personnel s'est plus concentrés sur les discussions avec le client au sujet de l'amélioration du scrapper générique et le choix des technologies ainsi que la compréhension de l'infrastructure de Cleep.

5.4.1 Scrapper générique

Le scrapper générique déjà présent n'était pas assez efficaces pour pouvoir enregistrer les données des produits et il était trop lent pour être utilisé en production indéfiniment (pour l'instant, si un lien n'appartient pas à la liste des 56 domaines scrappés, il passe par le scrapper générique). Comme il n'est pas non plus possible de scrapper de manière spécifique l'ensemble des sites commerciaux existants (la base de donées de Cleep compte plus de 8000 nom de domaines différents), j'ai discuté avec le client sur la nécessité d'un scrapper générique plus performant.
Ainsi en parallèle, j'ai essayé de developper un scrapper générique plus performant n'utilisant pas un "headless browser". La valeur ajouté de Cleep reposant sur les images du produit enregistré dans l'application, j'ai travaillé sur un scrapper générique permettant de récupérer les images du produit.
Ce scrapper suit les 2 étapes suivantes :

- Récupérations des liens "dynamique" de la page HTML
- Clustering des liens

Récupérations des liens "dynamique" de la page HTML:

Cette étape consiste à récupérer l'ensemble des liens images de la page n'étant des liens statique. Un lien image est considéré comme statique s'il est présent plusieurs fois sur différentes pages. En générale ces images sont les logos et autre widgets communément présents. L'exemple **figure 6** ci-après montre les images statiques sur le site de la fnac. Une fois connus, ces liens ne sont pas pris en compte par le scrapper générique.
Ainsi, à la fin de l'étape 1 (cf **figure 7** pour voir le procédé), l'ensemble des liens dynamique (ie liens d'images changeantes) est connu. Cependant, parmis ces images nous avons les images du produits, mais aussi celle fournies par les recommandations. Il faut donc trouver un moyen de séparer ces images .

Clustering des liens

Le but ici est de regrouper les liens de recommandations entre eux et les liens concernant le produits entre eux aussi. Pour cela j'ai utilisé un algorithme de clustering : dbscan. Le clustering se fait sur la distance en nombre de caractère dans le code HTML par rapport au début du code.
Cela permet donc d'avoir en général 2 cluster distincts. En me basant sur l'ensemble des sites que j'ai scrappé auparavant, le 1er cluster est choisi comme celui contenant les images que l'on souhaite. Ainsi à la fin de cette étape on est sensé avoir l'ensemble des images du produit. Afin de vérifier si les liens sont bien ceux recherchés, on compare les liens entre eux en utilisant la distance de Levenshtein [4]. La distance de Levenshtein

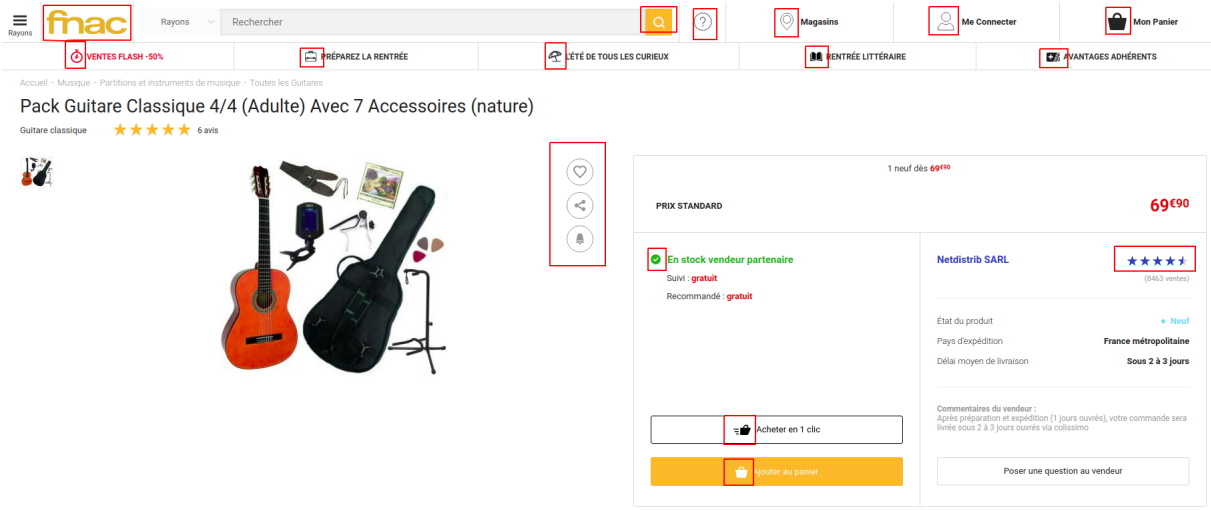

Figure 6: Images statiques

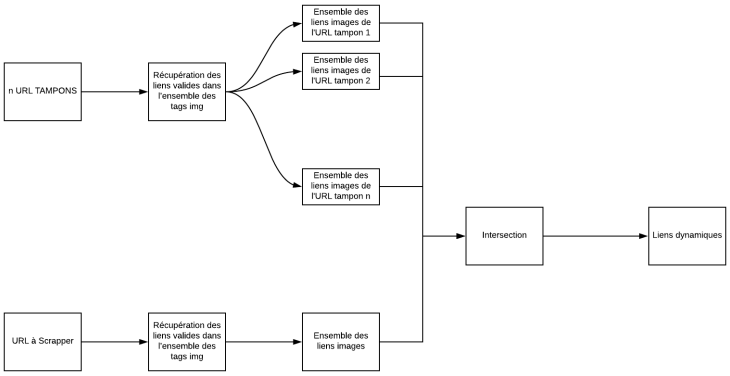

Figure 7: Etape 1

entre deux chaînes de caractère A et B est le nombre minimum de transformation (ajout/suppression/changement d'un caractère) pour passer de A à B. Ainsi, si dans le cluster, un lien paraît plus éloigné des autres, il est retiré de la liste.
Cette règle se base sur le fait que les images d'un même produits ont souvent des liens très similaires et sont différencié par un identifiant dans l'URL.

Tests et résultats

Les tests ont été réalisé sur une vingtaine de site que j'avais au préalable scrapper afin de pouvoir évaluer la différence. Il apparaît que ce scrapper générique arrive à récupérer toutes les images d'un produit spécifique. Cependant, il récupère trop d'image dans le sens où il peut récupérer 4 fois la même image mais ayant des tailles différentes. Encore une fois, pour des raisons de capacités mémoire, cette solution n'a pas été retenue par le client en l'état, mais est toujours en cours de développement.

5.4.2 Technologie utilisée

L'ensemble du travail a été réalisé en NodeJS (interpreteur Javascript). Cette technologie a été utilisé car l'ensemble de l'architecture client était en NodeJS. Il était possible de travailler avec les librairies Python permettant de scrapper et faire des requetes HTML (scrappy, beautifulsoup) car cette partie est plutôt indépendante. Cependant pour éviter l'overhead introduit par un changement de langage, j'ai préféré suivre le langage du client.
Par ailleurs, plusieurs sites internet ont des protections vis à vis du webscrapping. En effet les requêtes fait

aux différents serveurs affecte les capacités de ces derniers. Ainsi deux problèmes se sont posés:

- Ne pas lancer "involontairement" une attaque contre les serveur du site en question: pour cela on a trouvé une certaine limite temporelle avec le client
- Les adresses IP des serveurs de Cleep qui étaient déjà sur liste noire de plusieurs sites.

Le deuxième problème s'explique par une technologie de protection employé par les site internet contre le webscrapping : Datadome. Datadome detecte si une requête est faite par un utilisateur ou un robot en regardant si le javascript de la page est chargé et executé. Si ce dernier n'est pas executé, il s'agit probablement d'un bot ayant requêté uniquement le code source de la page, ce qui est exactement notre cas. Une fois detecté par Datadome, l'adresse IP qui a servi pour la requête est mise sur liste noire et il n'est plus possible de faire de requête.

La solution trouvé avec le client est de passer par un service de Proxy payant : Luminati, qui permet de changer d'adresse IP à chaque requête et donc de pouvoir utiliser les scripts de scrapping. Il est a noté que ce service facture à la requête, ainsi seul les sites ayant une protections datadome ou similaire passent par ce proxy.

6 Mission : Moteur de recommandation pour Cleep (2.5 mois)

Cette partie est consacrée au développement d'un moteur de recommandation pour la start-up Cleep. Cette mission fait directement suite à celle de webscrapping et reste dans l'esprit de développer des fonctions nécessaires à la croissance de l'application. Lors de cette mission, j'ai profité d'une grande autonomie car aucune solutions antérieures n'existaient auparavant. J'étais cependant en contact avec Damien Meurisse, co-fondateur de Cleep, et Paul Saffers, consultant datascientist à Sia Partners, afin de critiquer et valider mon avancement

6.1 Contexte

Le but de cette mission est de développer un moteur de recommandation : à partir d'un élément et des données qui lui sont liées, le moteur doit retourner un ou plusieurs éléments similaires. Cette pratique est vastement répandue parmi les sites marchands/ sites de services (Amazon, Netflix, Pinterest ...).
Avant le début de cette mission, Cleep n'avait pas de moteur de recommandation à proprement dit, certaines ébauches de moteur avait été faites, mais rien de concret. Pour une application comme Cleep dont le principe est de rendre l'utilisateur indépendant des sites marchands et de lui offrir une alternative au shopping basique sur Internet, la création d'un moteur de recommandation devenait nécessaire. En effet, il est possible de partager au travers de l'application les produits cleepés ainsi que les listes de cleeps. Un utilisateur peut voir les cleeps d'autres utilisateurs et en particuliers de ceux ayant les mêmes centres d'intérêts. Les interactions entre utilisateurs sont donc important, plus un utilisateur navigue sur Cleep, plus il passe de temps sur l'application ce qui le fidélise. Cependant sans moteur de recommandation, la navigation et la découverte de nouveaux cleeps n'était pas facilitée pour l'utilisateur. Celle-ci passait soit par la bar de recherche pour les cleeps spécifiques, soit par les pages catégories (triant les cleeps des utilisateurs par catégorie).
C'est donc dans l'optique de créer un moteur de recommandation fonctionnel et adapté à l'architecture déjà présente de Cleep que la mission m'a été confiée.

6.2 Objectifs et problématique

6.3 Travail effectué

6.4 Vision critique et apport personnel

7 Mission : Aide à l'amélioration d'une plateforme de déploiement pour Sia Partners

7.1 Présentation de la plateforme sialab

7.2 Objectifs et problématique

7.3 Travail effectué

7.4 Vision critique et apport personnel

8 Conclusion

9 Annexes

9.1 Code source allégé du site de la fnac

10 Glossaire

References

[1] Site de Cleep : https://app.cleep.io/

[2] Exemple de la Fnac : https://livre.fnac.com/a13195869/Agnes-Martin-Lugand-A-la-lumiere-du-petit-matin

[3] Exemple site internet avec plusieurs produits : https://livre.fnac.com

[4] Approfondissement sur la distance de Levenshtein : https://en.wikipedia.org/wiki/Levenshtein_distance

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

IMT Atlantique Bretagne - Pays de la Loire - www.imt-atlantique.fr

Campus de Brest
Technopôle Brest-Iroise
CS 83818
29238 Brest Cedex 03
T +33 (0)2 29 00 11 11
F +33 (0)2 29 00 10 00

Campus de Nantes
4, rue Alfred Kastler - La Chantrerie
CS 20722
44307 Nantes Cedex 03
T +33 (0)2 51 85 81 00
F +33 (0)2 51 85 81 99

Campus de Rennes
2, rue de la Châtaigneraie
CS 17607
35576 Cesson Sévigné Cedex
T +33 (0)2 99 12 70 00
F +33 (0)2 99 12 70 08