Big data management - Projet Lac de données

Lia Furtado and Hugo Vinson
Professor: J. Darmont

Université Lumière Lyon 2, France
lia.furtado@univ-lyon2.com
hugo.vinson@univ-lyon2.com

1 Introduction

Dans le cadre de l'enseignement Big Data Management du Master 2 Informatique mention Data Mining, nous allons mettre en place un Lac de données. L'objectif de ce travail est de préparer en utilisant diverses technologies le stockage de données et les méta-données associées. Le Data Lake est une méthode de stockage pour le Big Data qui se présente comme une alternative aux méthodes plus classiques tel que les Data Warehouse. Le Lac de données va permettre de stocker des données de structures, types et sources différentes et d'avoir des informations précises sur ces dernières à l'aide des méta-données associées. L'un des principaux avantages est la possibilité de stocker les données après chaque étapes des transformations, ainsi l'information est plus complète et permet de définir plus facilement de nouvelles utilisations à ces données. Pour ce projet nous allons travailler avec une source de données de type textuelle, il s'agit d'un corpus de texte que nous présenterons dans la section suivante. Une fois les données récupérées nous allons mettre en place un système de stockage de type système de fichier. Nous allons en suite construire et organiser le système de méta-données associé. Ce rapport présente dans un premier temps les données, puis nous décrirons le système de méta-données mis en place, feront par des analyses mené sur le corpus stocker au sein du Data Lake et enfin nous discuterons des différents points sensibles de nos travaux avant de conclure.

Tout le code pour reproduire les expériences de ce travail est sur le lien github **https://github.com/liasucf/ProjetDataLake**.

2 Data

2.1 Source

Les données représentent une partie des articles présent sur le site Wikipédia anglais[1] . Le corpus est disponible via le site Kaggle à l'adresse suivante : https://www.kaggle.com/ltcmdrdata/plain-text-wikipedia-202011. La taille du fichier à télécharger est environ 8Go, il s'agit d'une archive au format ZIP qui devra être décompressée pour accéder aux fichiers. Une fois décompressé, nous

[1] https://en.wikipedia.org/wiki/Main_Page

avons accès à un ensemble de fichier JSON, chaque fichier mesure environ 40 MB et contient une liste d'article wikipédia2, les articles ne semble pas avoir de lien direct entre eux. Pour chaque article nous avons accès à 3 informations:

- **id**: un entier unique pour chaque article de tous les fichiers
- **text**: une chaîne de caractère qui contient le contenu de l'article qui a été "aplatie"
- **title**: une chaîne de caractères qui représente le titre de l'article


```
[
 {
  "id": "17279752",
  "text": "Hawthorne Road was a cricket and football ground in Bootle in England...",
  "title": "Hawthorne Road"
 }
]
```


Fig. 1: Exemple d'un article issue du fichier source

2.2 Stockage

Le stockage des données peut se faire via différentes technologies lors du montage d'un Data Lake. Ici les données sont organisées sous la forme de clé-valeur, le choix le plus direct est alors de s'organiser autour d'un stockage de type NO-SQL. Suivant la consigne, nous avons décidé d'utiliser le système de gestion de fichier de Windows pour stocker les différents articles. Afin de faciliter le stockage nous avons créé un fichier pour chaque article, pour cela nous avons créé un script python pour automatiser le processus.

2.3 Traitement

Dans chaque documents, nous avions plusieurs articles, cette structure n'était cependant pas idéale pour extraire certaines métadonnées importantes, nous avons donc décidé de séparer chaque article dans un fichier séparé pour une meilleure cartographie des documents. Pour nos expériences, nous avons décidé de n'utiliser qu'un sous-ensemble des données afin de réduire le temps de calcul. Notre sous-ensemble avait 1447 documents où chaque document n'avait qu'un seul article.

3 Metadata

Les méta-données permettent d'avoir différents niveaux d'informations sur les données du Data Lake. Il existe différentes méthodes pour les classer et les organiser, nous avons décidé de suivre la méthodologie proposée par Sawadogo et Darmont[1]. Leur méthode s'organise en trois axes que nous présenterons succinctement. Tous d'abord les méta-données intra-objet, elles permettent d'avoir des informations sur la structure de l'objet, par exemple le nom du fichier, sa

taille, sa date de création mais aussi sur les données elles mêmes: le titre, le résumé... et enfin sur les traitements qu'elles ont subies: lemmatisation, embedding...
Le second type de méta-données correspond aux méta-données inter-objet: ici ce sont les liens entre les différents objets qui sont stockés. On retrouve deux types de liens, les liens physique : par exemple les liens de l'architecture du système de fichier ou le type de fichiers et les liens logiques: des liens sur le contenues des données, des champs partagés ou des similarités entre les textes... Enfin les méta-données globale, ici ce sont des méta-données qui vont permettre de construire des mesures par exemple un dictionnaire de synonyme, les auteurs y inclus également les index inversés.

Nous avons donc décidé de créer différentes méta-données que nous avons organisé de sorte à correspondre au modèle ci dessus. Nous allons dans cette partie présenter les différentes méta-données, la façon dont elles ont été créer et l'endroit où elles sont stockées.

3.1 Méta-donnée Intra-objet

Les méta-data concernant les objets peuvent être organiser en trois parties : les propriétés, cette partie contient des informations sur le stockage et le contenue de l'objet. La pré-visualisation: qui va permettre d'avoir une vue synthétique de l'objet. La version: l'objectif est de comprendre qu'elle traitement a subit la donnée. La construction des méta-données pour chaque objet a été faîte en Python et organisée sous forme d'un dictionnaire. Pour le stockage nous avons décidé de nous orienter sur du MongoDB [2] Il s'agissait de la méthode qui correspondait le mieux à nos données. Pour chacun de nos trois transformations du jeux de donnée, nous avons construit une collection, ce qui permet d'organiser les méta-données de tel à sorte à les séparer en fonction de la version. Les trois collections créée sont:

- intra_metadata_raw
- intra_metadata_cleaned
- intra_metadata_tfidf

Dans chaque collection nous retrouvons un dictionnaire qui décrit les méta-données de chaque fichiers. Nous avons détaillé ci-dessous les différentes méta-données que nous avons utilisée:

Propriétés

- file_name: Nom du fichier
- file_size: Taille du fichier
- creation_date: Date de création du fichier

[2] https://www.mongodb.com/

– sensitivity_level: Sensibilité des données
– title: Titre du texte
– document_type: Type du fichier
– language: Langage
– number_of_word: Nombre de mot

Pré-visualisation des données

– keywords: 10 mots les plus récurrents

Version

– transformation : Transformation faite sur le document (original / lemmatization)
– presentation : Structure du format du document (classique / tfidf)

Nous avons rempli chacune de ces informations pour chaque documents des collections. Dans la figure 2, nous pouvons voir comment ces informations sont structurées dans nos collections MongoDB pour un accès et une compréhension facile.

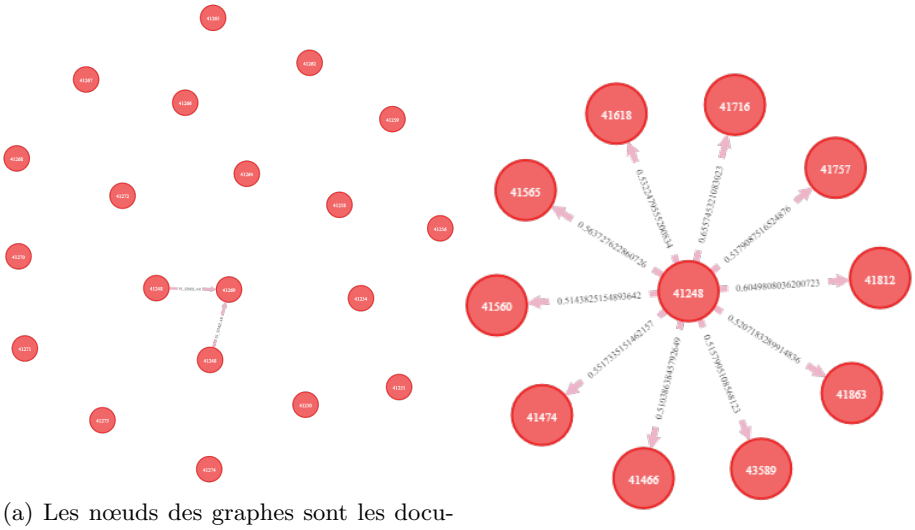
Fig. 2: Structure des intra-métadonnées dans une collection MongoDB

3.2 Méta-données Inter-objet

Les interactions entre les différents objets du Lac de données peuvent être de nature différente. Nous retrouvons dans un premier temps les liens physiques

entre les objets, ils font références aux liens directs, matériels entre deux objets, cela peut être un lien dans la structure de stockage ou le type de fichier auquel ils appartiennent. Le second type de liens sont les liens logiques, ici les liens sont de l'ordre du conceptuel, cela peut-être des catégories d'appartenance issues des données. Dans notre cas nous avons construit uniquement des liens logiques.

Lien logique Nous avons choisis de calculer la similarité cosinus entre les représentations TF-IDF de documents. La similarité cosinus est une mesure de la distance entre deux vecteurs. Afin de stocker ces liens, nous avons dû utiliser une structure permettant de cartographier ces relations entre ces documents. Nous avons utilisé Neo4j [3] pour construire la représentation. Dans ce graphique, chaque document est un nœud et les documents qui ont une similarité cosinus supérieure à 0,5 reçoivent une relation *IS_SIMILAR* Dans la figure 3a nous pouvons voir le graphe construit sur une partie des données. Les nœuds sont mappés avec l'identifiant du document et chaque relation a un poids défini par la valeur du cosinus (illustrée à la figure 3a).



(a) Les nœuds des graphes sont les documents et certains nœuds ont un rapport de similarité avec un autre document

(b) Les relations entre les documents pondérés par la similarité du cosinus

Fig. 3: Neo4j Graphiques

3.3 Méta-données Globale

Le Méta-données globale sont des méta-données qui fournissent des informations sur l'ensemble du Data Lake, on peut par exemple y retrouver des dictionnaires qui ont permis d'enrichir les données. Elles regroupent aussi les index qui seraient construits. Dans notre cas nous avons en premier temps un dictionnaire de correspondance entre les mots et le token associé lors de la Tokenisation.

[3] https://neo4j.com/

La Tokenisation à été faite en Python et nous avons donc le dictionnaire en mémoire. Nous avons décider de le stocker dans une nouvelle collection MongoDB étant donnée que la connexion était déjà ouverte, de plus si de nouveaux dictionnaire devait être charger nous pourrions facilement ajouter un champs dans cette collection pour le stocker. Dans la figure 4, nous voyons une partie du vocabulaire du tokenizer qui a été stocké dans une collection MongoDB appelée *global_metadata*.

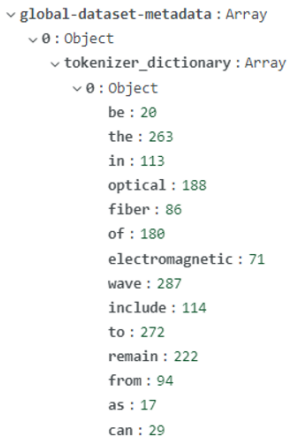


Fig. 4: Portion du vocabulaire du tokenizer tfidf stockée dans une collection MongoDB

Nous avons également mis en place un index inversé. Le principe de l'index inversé est de construire une table dans laquelle nous associons chaque mot du vocabulaire aux documents qui le possède. L'utilisation d'un tel index permet d'améliorer la recherche par mot clé. Pour le mettre en place nous avons utiliser Python, les données étant stockées sur un système de gestion de fichier nous pouvons facilement y avoir accès, de plus les traitements étant pour la plupart fait en python nous pouvons facilement intégré la création de l'index dans le pipeline. L'index est donc un dictionnaire python où chaque clé est un token associé à un mot et les valeurs sont des listes contenant le nom des fichiers qui ont ce mot dedans.Pour stocker l'index, nous avons ajouter un champs dans la collection *global_metadata*. Les requêtes MongoDB sont rapides en Python et elles permettront de récupérer la liste des fichiers qui possède le mot-clé. De plus MongoDB permet de faire des requêtes sur plusieurs collections, ainsi nous pourrions facilement avoir accès aux méta-données des fichiers associés à un mot-clé dans l'index inversé.

4 Analyses

Notre ensemble de données brutes sans transformation préalable était stocké dans un dossier appelé *enwiki-mini/* . Nous avions également un deuxième sous-ensemble de données qui contenait nos mêmes documents après avoir prétraité le texte. Dans chaque article, nous avons supprimer la ponctuation, les

stop-words , tous les caractères non latins, les chiffres et avons lemmatiser les mots. Ces données étaient stockées dans le système de fichiers Windows dans un dossier *enwiki-mini-clean/*. Et enfin, nous avons fait une autre transformation de données, nous avons pris les textes nettoyés et avons effectué une transformation TF-IDF. Nous avons choisi de limiter le vocabulaire TF-IDF à 500 mots afin de ne pas avoir de vecteurs de trop haute dimension. Ce troisième ensemble de données a été stocké dans un dossier *enwiki-mini-tfidf/*. Pour tout ce processus, nous utilisons le langage Python et des bibliothèques telles que pandas, numpy, spacy et sklearn.

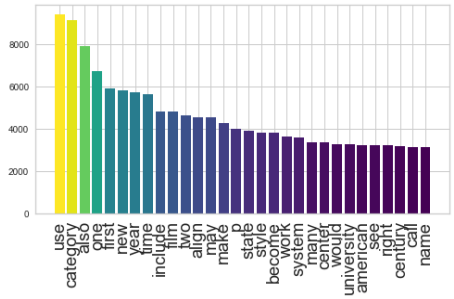
Afin d'analyser un peu les différences dans chaque ensemble de données de transformation, nous comparons les mots-clés du même article (id = 41248) dans la collection brute et nettoyée. Dans la figure 5 nous pouvons voir que les dix mots qui apparaissent le plus dans le texte au format brut sont principalement des ponctuations et des mots qui n'ont pas de signification sémantique réelle. En revanche, dans le même exemple, avec le format transformé après le processus de nettoyage, nous avons finalement quelques mots utiles qui nous permettent de mieux comprendre le sujet d'un texte.

(a) Mots clés du document brut (b) Mots clés du document nettoyé

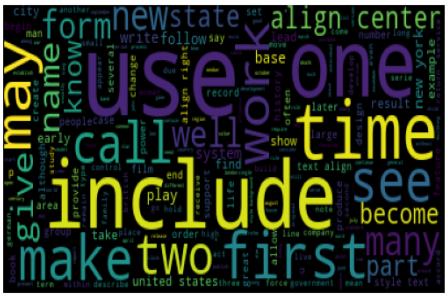
Fig. 5: Comparaison des mots-clés des collections brutes et nettoyées de MongoDB

De plus, pour mieux comprendre les textes et leur contenu, nous avons effectué une analyse exploratoire et tracé des graphiques usuelles en analyse textuelle. Pour construire ces graphiques, nous avons utilisé les documents textuels après le processus de nettoyage afin de pouvoir capturer la sémantique des documents. Dans la figure 6a, nous avons le graphique à barres qui montre la fréquence des 30 mots les plus utilisés dans tous les documents. À droite (Figure 6b), nous avons le nuage de mots qui est une représentation des mots : plus un mot est

présent dans le texte pris en considération, plus il apparaît en gros dans le nuage de mots. Nous pouvons constater que même si nous avons supprimé certains mots d'arrêt (par exemple and, the, to, was), il reste encore des mots qui sont plutôt génériques et qui ne disent pas grand-chose sur le contenu de l'article.



(a) Diagramme en bâtons des 30 mots les plus utilisés dans les documents avec leur fréquence.



(b) WordCloud avec tous les mots des documents

Fig. 6: Représentations de l'analyse textuelle

5 Conclusion

La création de ce Lac de donnée a été pour nous un véritable enjeux. Tout d'abord il s'agissait de travailler avec un gros volume de donnée. Nous avons donc dut mettre en place des stratégies pouvoir réaliser les traitements dans un temps raisonnable. Ensuite il s'agissait de créer un modèle de méta-donnée cohérent, qui permet d'avoir suffisamment d'information sur les données pour comprendre rapidement la manière dont elles s'agencent et d'où elles proviennent. Pour cela, nous avons utiliser une méthodologie développée dans la littérature sur le sujet. Enfin le troisième axe de complexité était l'utilisation de divers logiciels avec lesquels nous n'étions par forcément familier notamment : Noe4j. Pour conclure ce travail était très différent de ce à quoi nous étions habitué durant le Master et c'est avec beaucoup d'intérêt que nous l'avons mené. Devoir intervenir tout au long du cycle des données, de la récupération, la modélisation, le stockage, l'analyse jusqu'à la visualisation, nous a permis de bien comprendre les différents points sensibles qui surviennent lors de la création d'un Lac de données et de son utilisation. De plus nous avons pu nous rendre compte de l'intérêt du Lac de données par rapport aux entrepôts de données plus classique, l'historisation des traitements permet de pouvoir avoir la donnée sous sa forme brute ou prétraité ce qui est intéressant pour un travail de Data-Science. Enfin le gros travail effectué sur les méta-données nous as permis de mieux appréhender les différentes caractéristiques des objets et les différentes natures de lien qui les composes.

References

1. Sawadogo, P.N., Kibata, T., Darmont, J.: Metadata management for textual documents in data lakes (02 2019). https://doi.org/10.5220/0007706300720083