Certification Analyse de données massives
Projet NFE204

Analyse prédictive sur le site BoredPanda.com

Guillaume Payen
g1payen@gmail.com

Projet encadré par
Mr Philippe Rigaux

Promotion CNAM 2015 - 2016

Contents

1 Présentation

1.1 Analyse de données et choix du jeu données

1.1.1 Connaissances métier

Lorsque l'on fait de l'analyse de données, le travail sur la donnée (la préparation des données, la construction des modèles) est aussi important que le choix cohérent des variables qui serviront à l'analyse. Ce choix des variables ne peut se faire que sur une connaissance métier des données, et cette connaissance métier se forge sur l'expérience acquise sur le contexte d'utilisation de ces données.

C'est la raison pour laquelle pour réaliser une analyse pertinente, il faut connaître au préalable les données. Comment récupère-t-on les données? Sur quels critères sélectionne-t-on une variable plutôt qu'une autre? J'ai acquis toute mon expérience dans le secteur de l'Internet. Je suis donc particulièrement sensibilisé aux problématiques de marketing digital. C'est donc naturellement que j'ai choisi un jeu de données issu d'Internet, et qui puisse être utilisé pour réaliser une analyse marketing.

1.1.2 Présentation du projet d'analyse de données

Le site boredpanda.com est un site Internet de tendances de mode, de design, et de Buzz. Ce site a la particularité de générer beaucoup de trafic avec plus de 25 millions de visiteurs par mois. De plus, le contenu est en général repris sur beaucoup d'autres sites, ce qui fait de boredpanda.com une référence établie dans l'environnement buzz marketing. Quelques raisons a priori de ce succès:

- Les articles sont bien écrits
- Les articles comportent beaucoup d'images et de vidéos
- Les articles traitent toujours de thématiques très virales

Cet état de fait n'a en réalité pas beaucoup de valeur. Ce qui serait intéressant, c'est de savoir précisément quels sont les facteurs qui donnent de la popularité à un article plutôt qu'à un autre. Le fait est que les articles écrits sur boredpanda.com n'ont pas tous la même performance.

Le but de cette analyse est donc de mettre en évidence les caractéristiques des articles qui ont un impact sur leur popularité (et donc le trafic qu'ils génèrent). Cette analyse sera aussi prédictive, afin d'anticiper la popularité future d'un article avant même que ce dernier soit en ligne.

1.1.3 Intérêt de l'analyse d'un point de vue business

Boredpanda.com est un site gratuit. Pour autant, pour vivre, il a des frais de fonctionnement, comme l'hébergement du site, la rémunération des rédacteurs, les charges... Bien que les articles soient en consultation gratuite, le site doit donc bien générer de l'argent. Une façon pour ces sites de se rentabiliser est la publicité. En l'occurence, boredpanda.com utilise *Amazon Services LLC Associates Program*[1].

Il est donc vital pour le site de générer du trafic. Ils sont d'ailleurs en phase de recherche de community-managers, et de personnes qualifiées en marketing, afin d'augmenter la rétention des internautes, et le taux de conversion des articles.

Une analyse prédictive sur le contenu des articles du site boredpanda.com pourrait donner des indicateurs précieux aux responsables marketing afin de valider et dépasser leurs objectifs. L'enjeu est le maintien de l'activité, voire son développement, dans un secteur où il est extrêmement difficile de percer et développer son business à l'aide de la publicité.

[1] https://affiliate-program.amazon.com/

1.2 Sélection des outils nécessaires à la réalisation du projet

1.2.1 Programmation du projet avec NodeJS

La récupération de la donnée va être plus difficile que de simplement télécharger un fichier CSV ou JSON avec toutes les données à l'intérieur. Dans le cadre de cette analyse, il faut récupérer les données directement issues des articles du blog.

Il va donc être nécessaire de développer un programme qui soit capable de récupérer une liste d'articles, leur contenu, puis parser chaque article pour en extraire les variables qui composeront notre analyse. Deux langages ont été choisis : Python et NodeJS. Ces deux langages permettent de travailler au coeur de la donnée, et Python est déjà couramment utilisée pour faire de l'analyse de données. Cependant, à cette étape, le besoin n'est pas tant d'analyser la donnée mais d'extraire l'information. Et quoi de mieux pour travailler sur des données Web que d'utiliser des langages qui excellent dans le développement Web?

Voici donc pourquoi NodeJS a été choisi comme langage qui va prendre en charge toute la phase préparatoire de l'analyse, c'est à dire sa récupération, son nettoyage, son tri et son stockage dans une base de données NoSQL. Les raisons de ce choix sont les suivantes :

- NodeJS permet de travailler de façon asynchrone. Dans un contexte de workflow (ce qui est le cas ici), NodeJS se révèle donc beaucoup plus rapide que Python
- NodeJS permet d'allier traitements backend, et visualisation frontend sur un site Internet, ce qui le rend polyvalent et clairement indiqué pour faire du monitoring sur les données collectées
- Il existe des packages NodeJS permettant de parser du code HTML comme il est possible de le faire avec JQuery[2], référence en développement FrontEnd pour interagir avec le rendu HTML du serveur
- Des packages NodeJS permettent de réaliser des cronJobs avec programmation par intervalle, et programmation à date fixe

1.2.2 Cassandra, base de données NoSQL

Le choix de la base de données NoSQL pour le stockage des données s'est fait selon plusieurs critères:

- Le système est-il adapté à de l'analyse de données dans un contexte big data?
- Le système est-il adapté à un environnement distribué?
- Le système est-il assez mature?
- Le système possède-t-il des connecteurs NodeJS ?

De cette recherche, il ressort que Cassandra semble être le moteur de bases de données idéal pour ce projet d'analyse de données. En effet, il exite depuis 2008 et est utilisé par de grandes entreprises dans des environnements de données massives. Cassandra possède des connecteurs NodeJS, et également des connecteurs Spark. Ainsi, il est possible de stocker les données extraites du site boredpanda.com, et faire de l'analyse prédictive avec Spark et ses librairies de machine learning telles que MlLib. Enfin, le système de distributivité chez Cassandra est assez remarquable et en fait un système de bases de données de choix pour faire de la réplication et du partitionnement.

[2] https://jquery.com/

1.2.3 Virtualisation des environnements avec Docker

Docker[3] est devenu en quelques années une référence dans la virtualisation d'environnements. L'utilisation de Docker dans le contexte de ce projet est particulièrement indiqué pour les raisons suivantes :

- En utilisant Docker, il est possible d'empaqueter le projet afin de le déployer facilement, sur différents environnements sans les contraintes d'administration du système
- En utilisant Docker, il est possible aux lecteurs intéressés par cette analyse de récupérer le container du projet, et de le faire tourner sur leur machine, afin de voir comment la donnée s'agrège
- La donnée étant stockée sur Cassandra, l'utilisation de Docker rendra plus facile les utilisations pratiques de Cassandra en mode cluster, avec tests de réplication. Sur une seule machine, il sera alors possible de déployer autant de noeuds que souhaité.

[3] https://docker.com

2 Sélection des variables explicatives

2.1 Utilisation des connaissances métier

C'est ici que la connaissance métier du domaine peut faire la différence dans le cadre de cette analyse de données. En effet, la donnée extraite de l'article est initialement son code source. Il faut donc véritablement fouiller dans le code source de chaque article pour en retirer les informations qui seront pertinentes. Ces informations prises indépendamment les unes des autres peuvent sembler anodines, et pourtant, considérées dans leur ensemble, elles peuvent apporter du sens.

L'extraction de l'information va s'axer selon 4 axes qui sont - d'un point de vue marketing - les axes les plus importants susceptibles d'avoir un impact sur le comportement des internautes.

2.2 Présentation des variables explicatives

2.2.1 Variables liées au contenu textuel de l'article

La première catégorie de variables traite du contenu brut des articles, de la façon dont ils sont rédigés. Cette catégorie de variables prend en compte les données textuelles qui composent un article.

Ces variables vont donc rendre compte des occurrences de différentes balises comme les titres, le nombre de liens, la taille du titre de l'article, etc. Le but au travers de ce type de variables est d'avoir une vision quantitative de l'article, en utilisant des caractéristiques de l'article qui a priori n'apporteraient pas grand chose à sa popularité, mais qui, considérées dans leur ensemble pourraient faire émerger des motifs de rédaction qui ont un impact sur le ressenti de l'Internaute, et donc son envie de lire l'article.
La liste complète et expliquée de ces variables se situe en annexe A.

2.2.2 Variables liées aux catégories liées à l'article

La seconde catégorie de variables traite de la classification des articles en catégories. Sémantiquement parlant, il est tout à fait acceptable de considérer que certaines catégories sont plus populaires que d'autres. Il est donc important de pouvoir analyser l'impact de ces catégories sur le succès d'un article.

Préalablement ont été extraites toutes les catégories du site (45). En se servant des *tags* de chaque article, il est alors possible de réaliser leurs calculs de similarité avec les catégories pré-définies. Un article peut appartenir à plusieurs catégories différentes, et on compte donc 1 variable différente par article, avec pour chaque variable une valeur booléenne caractérisant l'appartenance de l'article à cette catégorie.

La liste complète et expliquée de ces variables se situe en annexe B.

2.2.3 Variables liées au contenu média de l'article

La troisième catégorie de variable concerne les médias insérés dans l'article. Il a été prouvé d'un point de vue marketing qu'un document contenant des images ou des vidéos a plus d'impact sur le lecteur. On parle ici d'une tendance qui est propre au contenu rédigé en ligne, qui a vu le jour il y a moins de 10 ans et qui a fondamentalement modifié la relation de l'internaute avec le contenu. Certains articles sont maintenant rédigés avec - pour seul contenu - un diaporama, sans autre mention de texte. En l'occurrence, boredpanda.com utilise aussi cette pratique dans nombre de ses article. Ces variables caractérisent donc le nombre d'images, leur forme horizontale ou verticale, le nombre de vidéos, etc.

La liste complète et expliquée de ces variables se situe en annexe C.

2.2.4 Variables liées à l'auteur de l'article

La quatrième catégorie de variables s'intéresse à l'auteur de l'article. La popularité d'un article peut tout autant dépendre de son contenu que de celui qui l'a rédigé. Certains auteurs sont très populaires sur Internet, au point que pour certains, leurs rédactions sont décrétées *à succès*, et génèrent énormément de trafic, que le contenu soit qualitatif ou non. L'idée est donc de créer des variables qui caractérisent l'auteur de l'article, comme s'il a une page Facebook, un site web, s'il fait partie du staff de boredpanda.com, etc.

La liste complète et expliquée de ces variables se situe en annexe D.

2.3 Présentation des variables métriques

Il existe plusieurs variables métriques qui représentent directement la popularité d'un article. Il s'agit du nombre de vues de l'article, et du nombre de partages sur Facebook. Ces deux informations sont écrites en clair dans le code source de l'article, et il est vital de pouvoir les utiliser. Un des challenges de l'analyse sera de choisir LA variable quantitative à expliquer parmi les deux, et également savoir si la variable restante peut être remise dans le lot de variables explicatives, ou est-ce qu'il faut la rejeter du fait d'un coefficient de corrélation trop important.

La liste complète et expliquée de ces variables se situe en annexe E.

3 Procédure de collecte des données

3.1 Présentation des contraintes temporelles

Alors que se met en place la stratégie de collecte de la donnée, apparaît tout de suite un premier obstacle de fond. Lorsque les articles sont créés, ils ont par définition un nombre de vues qui est nul. Si l'on récupère l'article au moment de sa création alors il ne sera d'aucune utilité car la variable quantitative à expliquer sera inexploitable. De même, le nombre de vues d'un article va - au fil du temps - devenir de plus en plus grand. Si l'on considère le même article à J+1 et à J+10, son nombre de vues sera clairement différent. Il n'est donc pas possible de récupérer les articles à la volée, car pris à des moments différents de leur *vie*, les articles n'auront rien à voir et ne seront pas comparables.

On a donc ici une réelle contrainte temporelle qui nous oblige à considérer - pour pouvoir réaliser une analyse de données correcte - des articles extraits après un délai fixe du moment de leur mise en ligne. Ce délai doit être incompressiblement le même pour chaque article, et doit être assez grand pour pouvoir recueillir un nombre de vues significatif.

D'un point de vue marketing, un article se comporte un peu comme un produit, et il se trouve qu'il suit lui aussi le cycle d'engagement de Gartner[4], qui se formalise ainsi :
L'engouement de l'article est donc très élevé lorsqu'il vient d'être mis en ligne, puis l'engouement

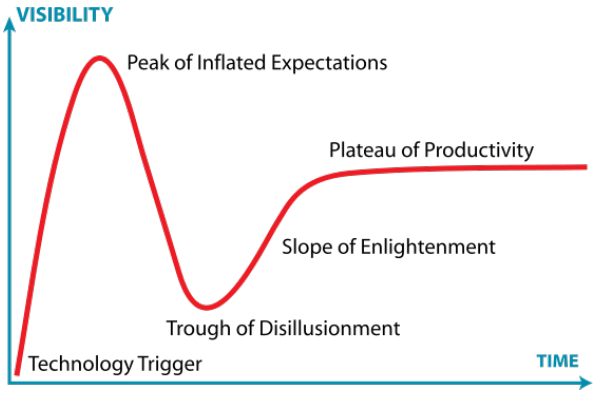

Figure 1: Cycle de Gartner

retombe ensuite assez rapidement. Après avoir fait quelques essais, il s'est avéré à sa mise en ligne, le nombre de vues d'un article augmente de façon fulgurante. Puis, au bout de quelques jours, il tend à stagner sans pour autant jamais s'arrêter.

Dans le cadre de ce projet, il sera choisi de procéder ainsi. Lorsqu'un article est mis en ligne, l'article va être récupéré mais pas analysé. Seule son URL sera extraite et renvoyée à un module de NodeJS qui fait office de cronjob, lui demandant de réaliser l'analyse du document dans exactement 3 jours. Les articles sont ainsi analysés en deux passes. Le premier accès à l'article sert de routage temporel, et le second accès au même article 3 jours plus tard conduira à l'extraction des variables.

En procédant ainsi, on s'assure que :

- Les articles ont généré un nombre de vues significatif
- Les articles sont tous analysés précisément au même moment après leur mise en ligne

[4] https://en.wikipedia.org/wiki/Hype_cycle

3.2 Récupération de la liste des articles

L'objectif de cette étape est de récupérer les articles dès qu'ils viennent d'être mis en ligne, afin de les router vers le cronjob. En règle générale, les blogs disposent d'un flux RSS, et il est alors possible de récupérer le flux. Comme le flux est mis à jour dès qu'un article est mis en ligne, il est alors possible de récupérer en temps réel les articles mis en ligne.

Le fait est que boredpanda.com ne dispose pas de flux RSS. De plus, son compte Twitter n'est pas alimenté proprement, certains articles n'y figurent pas. Impossible donc d'utiliser Twitter pour récupérer l'adresse des articles qui viennent d'être mis en ligne.

Heureusement, grâce à Google il existe une parade. Google a littéralement phagocyté la recherche sur Internet, et a par la même occasion imposé ses normes de SEO[5]. Afin de mieux les référencer, Google demande donc aux sites Internet de notamment lui fournir - au format XML - un fichier contenant toutes les urls du site. Ce fichier, appelé *sitemap* est presque tout le temps appelé *sitemap.xml*, et est à la racine du site.

Une brève analyse de la structure du site boredpanda.com révèle que ce dernier a été réalisé avec Wordpress. Dans ces conditions, la sitemap[6] a sûrement été générée par un plugin, ce qui est un gage de sécurité pour l'extraction des documents, car c'est la garantie que le ficher aura toujours la même forme, et que sa mise à jour sera programmée. Le serveur NodeJS va donc se connecter à la sitemap à raison de 1 à 2 fois par jour, récupérer la liste des nouveaux articles postés sur le site, rajouter à leur date de création respective une durée de 3 jours, et va router l'article et la date calculée au cronjob NodeJS.

Il s'avère inutile d'interroger le site toutes les heures par exemple, pour les raisons suivantes :

- Autant de connexions répétées régulières pourraient sembler suspectes pour les administrateurs du site, et il ne faudrait pas qu'ils bannissent l'IP de notre service
- Même si on récupère un article écrit la veille, la durée de 3 jours est ajoutée à la date de création de l'article contenue dans la sitemap, non la date à laquelle la sitemap est récupérée. L'heure où on récupère la sitemap est indifférente, mais il faut que la récupération de cette liste se fasse avant que l'article ait été créé depuis 3 jours.

3.3 Récupération de l'article et extraction de l'information

Lorsque l'article a été routé dans le cronjob, il reste en attente pendant une durée de 3 jours à partir de sa mise en ligne. Après 72 heures, il est analysé par le serveur NodeJS. Ce dernier va utiliser l'URL de l'article pour en récupérer son code source, à l'aide d'une requête assimilable à du CURL.

NodeJS va utiliser un module qui va permettre de parser le code HTML de l'article de la même façon que le fait JQuery sur du FrontEnd. De cette analyse, il sera alors possible d'extraire toutes les variables textuelles, catégorielles, et médias. L'URL de l'auteur de l'article sera également récupérée.

Une seconde requête sera engagée, cette fois-ci sur l'URL de l'auteur afin de récupérer le code source de sa page profil sur boredpanda.com. A l'aide des même outils, NodeJS va extraire toutes les variables liées à l'auteur.

L'ensemble de toutes ces informations seront regroupées dans un seul et unique document au format JSON.

[5] Search Engine Optimization
[6] http://www.boredpanda.com/sitemap.xml

3.4 Sauvegarde de la donnée dans Cassandra

Le connecteur Cassandra de NodeJS va être utilisé pour sauvegarder le document dans la base de données. Étant donné que les documents stockés auront tous la même structure et représentent la même information, il sera possible de n'utiliser qu'un seul *keyspace*.

Le schéma présentant présentant la structure des rowkeys, et le typage des *columns* se situe en annexe F. Le choix a été fait de regrouper les variables par famille plutôt que de toutes les stocker dans un champ différent. Ce choix a été fait pour plusieurs raisons:

- Cassandra permet de stocker de l'information de type *map*
- En regroupant l'information par type de variables, il apparaît un vrai gain de lisibilité en lecture, et une simplicité de traitements en écriture
- Le schéma reste souple, et il sera possible d'ajouter de nouvelles variables dans une catégorie de variables, sans mettre à jour le schéma de la base de données
- Dans le cadre d'une analyse de données, les données seront récupérées en bloc. Il n'y a donc pas de besoin de pouvoir faire des requêtes spécifiques sur une variable ou une autre

Le stockage d'un article dans Cassandra se présentera ainsi de la manière suivante :

```
cqlsh:boredpanda> select * from data where
id='http://www.boredpanda.com/refugee-life-jackets-konzerthaus-ai-weiwei/';

id             | http://www.boredpanda.com/refugee-life-jackets-konzerthaus-ai-weiwei/
author_info    | {'baseline': 4, 'baselineRatio': 0.556, 'staff': 1}
author_links   | {'etsy': False, 'facebook': False, 'flickr': False, 'imgur': False,
                 'instagram': False, 'site': False, 'tumblr': False, 'twitter': False}
author_metrics | {'featuredPots': 75, 'likesPerPost': 363, 'pointsPerPost': 30,
                 'posts': 80, 'viewsPerPost': 3400, 'viewsTotal': 21700000}
category       | {'Advertising': False, 'Animals': False, 'Architecture': False, 'Art': False,
                 'Automotive': False, 'Body Art': False, 'Challenge': False, 'Comics': False,
                 'DIY': False, 'Digital Art': False, 'Drawing': False, 'Entertainment': False,
                 'Food Art': False, 'Funny': False, 'Furniture Design': False, 'Good News': False,
                 'Graphic Design': False, 'History': False, 'Home': False, 'Illustration': False,
                 'Installation': False, 'Interior Design': False, 'Land Art': False,
                 'Nature': False, 'Needle and Thread': False, 'Optical Illusions': False,
                 'Other': False, 'Packaging': False, 'Painting': False, 'Paper Art': False,
                 'Parenting': False, 'Photography': False, 'Pics': False, 'Product Design': False,
                 'Recycling': False, 'Science': False, 'Sculpting': False,
                 'Social Issues': False, 'Street Art': False, 'Style': False, 'Technology': False,
                 'Travel': False, 'Typography': False, 'Video': False, 'Weird': False}
comments       | {'has_comments': True}
content        | {'ratio': 0.397, 'words': 144}
date           | {'day': 'Tuesday', 'weekend': 'No'}
description    | {'ratio': 0.412, 'words': 30}
h              | {'h1': 0, 'h2': 0, 'h3': 6, 'h4': 0, 'p': 22}
is_article     | True
media          | {'horizontal': 9, 'pictures': 9, 'ratioHorizontal': 1, 'ratioSquare': 0,
                 'ratioVertical': 0, 'square': 0, 'vertical': 0, 'video': 0}
metrics        | {'share': 644, 'views': 6600, 'vote': 135}
more           | {'etsy': False, 'facebook': True, 'flickr': False, 'imgur': False,
```

```
                  'instagram': True, 'site': True, 'tumblr': False, 'twitter': False}
tags              | {'nbTags': 12, 'tagLength': 12.333}
title             | {'ratio': 0.083, 'words': 11}
```

3.5 Schéma global du processus d'extraction de la donnée

Les étapes de l'extraction des informations des articles du site boredpanda.com sont résumées dans ce schéma :

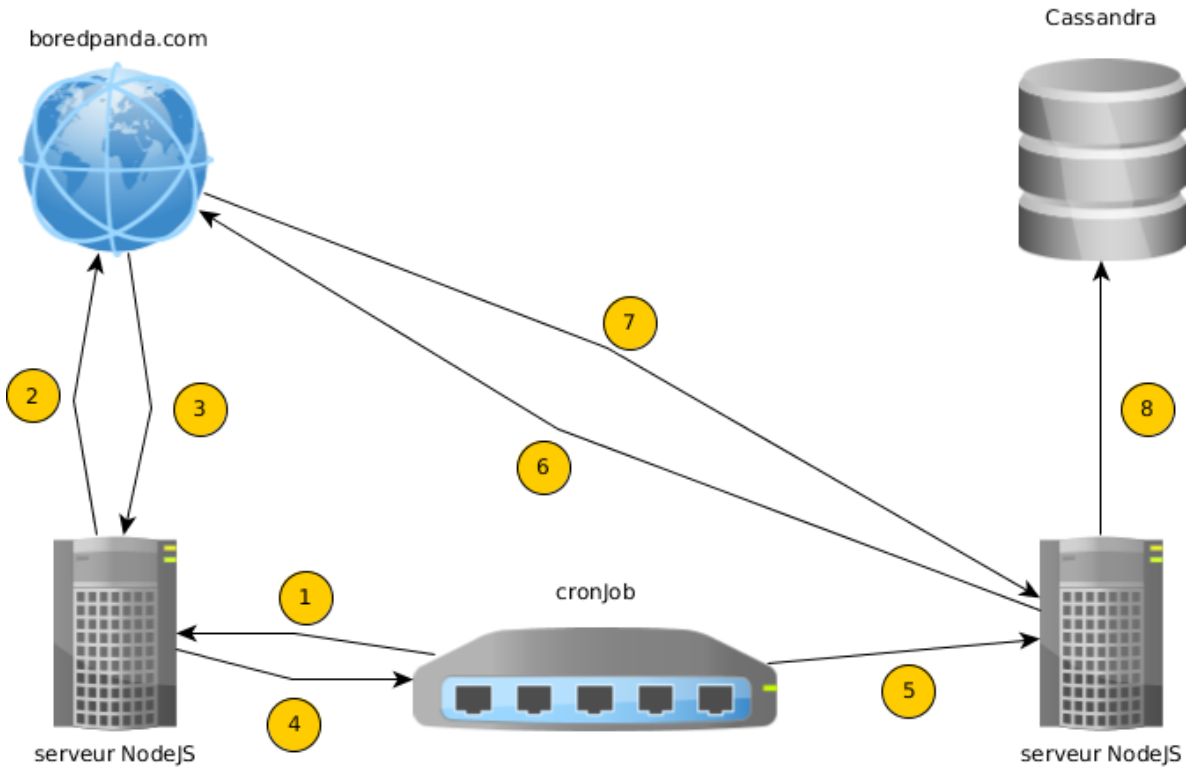

Figure 2: Schéma de l'extraction de l'information du site boredpanda.com

Pour des raisons de lisibilité du schéma, le même serveur NodeJS a été représenté 2 fois, mais il ne s'agit que d'une seule et unique instance.

1. 2 fois par jour, le cronJob appelle le serveur NodeJS
2. le serveur NodeJS fait une requête sur la sitemap *http://www.boredpanda.com/sitemap.xml*
3. la sitemap est renvoyée au format XML au serveur NodeJS
4. le serveur NodeJS extrait l'URL des nouveaux articles et les renvoie au cronJob où ils vont rester en attente
5. l'article reste en attente pendant 3 jours après sa création, ensuite son URL est renvoyée au serveur NodeJS
6. le serveur NodeJS fait plusieurs requêtes sur l'article pour récupérer son contenu et les informations de l'auteur

7. le site boredpanda.com renvoie à chaque fois la réponse aux requêtes, au format HTML

8. Après traitements et extraction, la donnée est sauvegardée dans une base de données Cassandra

Le processus de collecte des données ici présenté met en évidence un phénomène de latence assez important. En effet, la mise en production du projet a montré une agrégation de environ 10-15 articles par jour. On pourrait espérer commencer une analyse de données après au moins 2 ou 3 mois de collecte. Ainsi, avec quelques centaines d'observations, le choix des méthodes d'analyse des données serait plus confortable. Pour le cas des réseaux de neurones par exemple, il est nécessaire d'avoir un nombre d'observation assez grand pour que l'algorithme puisse produire des résultats concluants.

Au delà de la récolte relativement lente de l'information, il faut prendre en compte également que chaque article met au final plus de 3 jours à être inséré en base. Bien que robuste, un tel processus de collecte de l'information présente des risques qui seront discutés après.

4 Analyse du système de réplication chez Cassandra

4.1 Fonctionnement de Cassandra en cluster : le Hash-Ring

En mode cluster, Cassandra met en avant une architecture de serveurs bien particulière, qui distingue ce moteur de bases de données des autres protagonistes NoSQL, et qui en fait d'ailleurs un système extrêmement puissant.

L'architecture des noeuds avec Cassandra se fait au travers ce que qui est communément appelé un *Hash Ring*. Chaque noeud du cluster est disposé en file, dans un *cercle* directionnel. Lorsque l'on ajoute un nouveau noeud dans le cluster, ce dernier vient naturellement s'ajouter au Ring. C'est notamment à partir de cette caractéristique qu'une phrase est souvent reprise dans la littérature lorsqu'il s'agit de faire de la réplication avec Cassandra : *Just add a node !*

Chaque noeud a un token qui va permettre de représenter des intervalles d'affectation des documents sur un noeud particulier. On parle alors de consistent hashing car grâce à ce hashage, les documents sont uniformément répartis sur le Ring.

Le choix d'un noeud pour l'écriture dans le Hash Ring se fait grâce à des partitioners qui proposent des méthodes de hashage différentes, permettant une répartition uniforme des ressources dans le cluster, et évider ainsi l'apparition de hotspots. Nous n'irons pas plus loin dans ces explications, car elles traitent du partitionnement, et notre analyse va se concentrer sur la réplication.

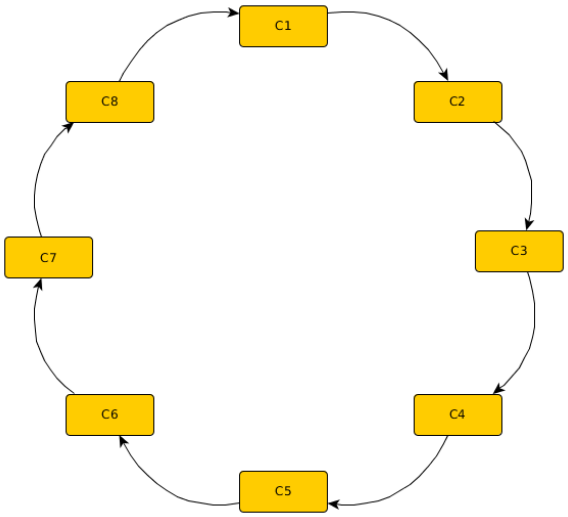

Figure 3: Représentation d'un cluster Cassandra avec le Hash Ring

4.2 Gestion des requêtes par Cassandra

Un cluster Cassandra a la particularité de fonctionner en mode *peer*. La notion de noeud maître et noeud esclave n'existe donc pas avec Cassandra. Chaque noeud du cluster a le même rôle et la même importance, et jouit donc de la capacité de lecture et d'écriture dans le cluster. Un noeud ne sera donc jamais préféré à un autre pour être interrogé par le client.

Pour que ce système fonctionne, chaque noeud du cluster a la connaissance de la topologie du Ring. Chaque noeud sait donc où sont les autres noeuds, quels sont leur token, quels noeuds sont disponibles

et lesquels ne le sont pas.

Du coup, lorsque un client interroge Cassandra, en mode cluster, il va interroger un noeud qui sera choisi au hasard parmi tous les noeuds du cluster. Évidemment, le partitionnement fait que tous les noeuds ne possèdent pas localement l'information recherchée. Cependant, tous les noeuds sont capables de dire quel est le noeud du cluster qui possède la ressource recherchée. Dans ce cas donc, ce noeud que l'on appellera *le coordinateur* va rediriger la requête au bon noeud, c'est à dire celui qui est capable de la traiter, et ce dernier renverra directement le résultat au client.

4.3 Facteur de réplication et stratégie de placement des réplicas

Le facteur de réplication est le paramètre du *Keyspace* qui précise le nombre de réplicats qui seront utilisés. Le facteur de réplication par défaut est de 1, signifiant que la ressource sera stockée sur un seul noeud. 3 est la valeur conventionnelle du facteur de réplication optimale pour assurer la disponibilité du système.

Le facteur de réplication est un indicateur qui précise le nombre final de copies de la ressources dans le cluster. Si le facteur est de 3, il ne sera donc pas écrit 1 fois, puis répliqué 3 fois, mais écrit 1 fois, et répliqué 2 fois.

Un autre paramètre lié aux keyspaces est le paramètre topologie de la réplication, c'est à dire la logique selon laquelle les ressources seront répliquées au sein des noeuds du cluster. 2 seront présentées ici, la stratégie simple, et la stratégie par topologie du réseau.

4.3.1 Stratégie simple

Avec la stratégie simple, tout part du Hash Ring. Considérons un cluster composé de 8 noeuds, c1 à c8, et un facteur de réplication de 3. Comme il a été expliqué précédemment, n'importe quel noeud peut recevoir la requête du client. Ce noeud, que l'on nommera *coordinateur* va utiliser la méthode de hashage, le token des noeuds du cluster et la clé de la ressource pour décider quel sera le noeud dans lequel cette dernière sera stockée. Le coordinateur va alors rediriger la requête pour une écriture sur le noeud choisi par la fonction de hashage. Comme le facteur de réplication est de 3, le coordinateur va aussi rediriger la requête d'écriture vers les 2 noeuds suivant le noeud choisi, dans le sens du Hash Ring
Comme on le voit dans le schéma ci-dessus, lorsque le client effectue la requête sur le cluster, c'est

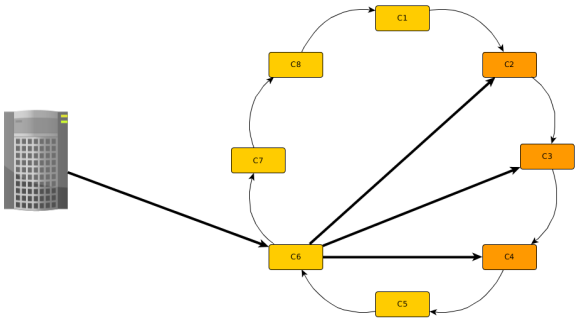

Figure 4: Stratégie de réplication simple

le noeud c6 qui a été sélectionné (au hasard) pour traiter la demande. Ce dernier calcule que c'est le noeud c2 qui doit être sollicité pour traiter la requête. Il va donc rediriger la requête vers c2, mais

également vers c3 et c4

Ce schéma vaut aussi pour la lecture que pour l'écriture.

4.3.2 Stratégie par topologie du réseau

La stratégie par topologie du réseau présente un intérêt lorsque l'infrastructure est répartie sur différents clusters. Ces derniers peuvent être loin géographiquement, ou peuvent être dans le même local. Avec cette stratégie, Cassandra va mettre en avant la localité des données, et préférera alors solliciter les noeuds locaux. C'est notamment la raison pour laquelle cette stratégie est intéressante pour des ressources localisées dans différents endroits du monde.

L'architecture est toujours celle d'un Hash Ring, chaque noeud étant lié au noeud suivant, et le Hash Ring est toujours directionnel. L'écriture d'un document va se faire pour chaque groupe de noeuds, selon le facteur de réplication. Comme on le voit dans le schéma ci-dessus, le système est

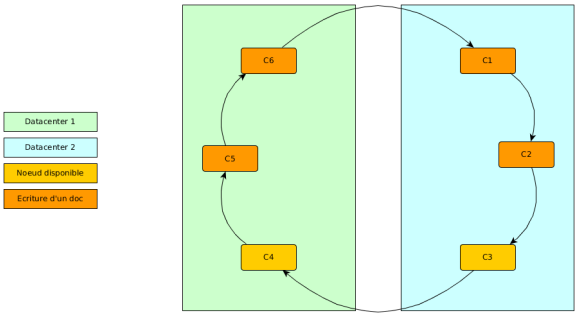

Figure 5: Stratégie de réplication par topologie du réseau

réparti sur 2 data centers, et le facteur de réplication est de 2. L'écriture des ressources va donc se faire sur chaque datacenter, et au sein de chaque datacenter, le facteur de réplication sera utilisé pour répliquer les données selon la direction du Hash Ring, de la même manière que dans une stratégie simple. Pour chaque data center donc, la ressource sera écrite sur le noeud qui fait correspondre son token avec le hash de la clé du document, par la fonction de hashage, et sur le noeud suivant.

4.4 Écriture et cohérence des données

La cohérence se réfère à la capacité d'une ressource d'être synchronisée et à jour parmi tous les réplicats. Avec Cassandra, la cohérence des données en écriture est paramétrable. Ce paramétrage est nécessaire pour affiner la stratégie à adopter en cas de problèmes d'écriture.

Considérons par exemple l'écriture sur un noeud inactif. Lorsque cela arrive, le coordinateur va attendre que le noeud soit de nouveau actif avant d'écrire la ressource sur ce dernier. Bien entendu, rien ne l'empêche d'écrire sur les autres réplicats, ce qu'il fait d'ailleurs. Lorsque le coordinateur attend la remise à disponibilité du noeud, il stocke alors la ressource localement. Pendant cette attente, le document n'est pas à proprement parler dans le cluster. Il n'est donc pas disponible à la lecture. Ce cas de figure s'appelle un *Hinted Handoff* et met en avant le problème lié à la cohérence des données

Différentes stratégies ont vu le jour pour gérer le Hinted Handoff. Elles vont de celles qui vont maximiser la disponibilité du système à celles qui vont maximiser la cohérence des données. Quelques stratégies sont résumées ci-dessous :

- ANY : La réponse au client sera assurée si la ressource a été écrite sur au moins 1 des réplicats. Si tous les noeuds sont inactifs, alors le Hinted Handoff est toléré. C'est la stratégie qui ne garantie pas la cohérence des données, mais qui rend le système le plus disponible. En l'occurrence, c'est aussi la stratégie la plus dangereuse, elle peut entraîner de nombreux conflits
- ONE : La réponse au client sera assurée si la ressource a été écrite sur au moins 1 des réplicats. Impossible ici d'utiliser le Hinted Handoff.
- QORUM : La réponse au client sera assurée si la ressource a été écrite sur au moins la moitié des réplicats. C'est aujourd'hui un très bon compromis, car la règle du quorum va s'adapter au nombre de réplicats considéré
- ALL : La réponse au client sera assurée lorsque la ressource aura été écrite dans tous les réplicats. C'est la stratégie qui assure la meilleure cohérence des données, au prix de la disponibilité

Comme dans d'autres systèmes de bases de données de type NoSQL, la stratégie à adopter pour assurer la cohérence des données en écriture est souvent affaire de compromis. Plus on fait en sorte que le système soit disponible, plus on l'expose à la création de conflits car en procédant ainsi, le système ne peut plus assurer la cohérence des données. A contrario, si on veut assurer la meilleure cohérence des données, alors il faut s'assurer pour chaque écriture que la ressource a été écrite partout, ce qui rend du coup le système beaucoup moins disponible.

4.5 Lecture et cohérence des données

Lorsque en lecture la ressource n'est pas synchronisée entre les différents réplicats, alors le système détecte un conflit. Lorsque le coordinateur identifie un conflit, il va solliciter les réplicats pour démarrer une procédure de réconciliation. La réconciliation peut démarrer, mais le rôle du coordinateur n'est pas d'attendre que la réconciliation soit terminée pour répondre au client. En l'occurrence, Cassandra garantit que au prochain appel, les données seront de nouveau synchronisée, mais pour l'appel présent, le système a besoin d'une stratégie pour savoir quelle ressource renvoyer au client.

Comme pour l'écriture de données, il existe aussi des stratégies de cohérence de données en lecture. Il y a des stratégie qui vont optimiser la réactivité du système, et donc sa disponibilité. D'autres stratégies, vont mettre en avant la vérification de la cohérence des ressources, au détriment de la disponibilité. Quelques stratégies sont résumées ci-dessous:

- ONE : Le coordinateur reçoit la réponse du réplicat le plus proche et la renvoie au client. Cette stratégie assure une haute disponibilité, mais au risque de renvoyer une ressource qui n'est pas synchronisée avec les autres réplicats. Dans ce cas, la cohérence des données n'est pas assurée
- QUORUM : Le coordinateur reçoit la réponse de au moins la moitié des réplicats, et renvoie au client la ressource avec le timestamp le plus récent. C'est la stratégie qui représente le meilleur compromis
- ALL : Le coordinateur reçoit la ressource de tous les réplicats. Si un réplicat ne répond pas, alors la requête sera en échec. C'est la stratégie qui assure la meilleure cohérence des données, mais au prix de la disponibilité du système

Pour la lecture aussi, la performance du système est affaire de compromis. Pour assurer une réponse qui reflète exactement les ressources stockées en base, il faut interroger plusieurs réplicats (voire tous), ce qui prend du temps. La disponibilité du système va donc être fortement dégradée. Si au contraire, on veut le système le plus disponible possible, alors il faut ne lire la ressource que sur 1 seul réplicat, et la renvoyer directement au client. Il faudra dans ce cas accepter qu'il n'est pas impossible que le client reçoive une ressource non synchronisée, et donc fausse.

4.6 Utilisation des snitches : Gossip

Les snitches sont des modules de Cassandra, qui occupent des fonctions très particulières. Dans le cadre de cette analyse, nous ne considéreront que le Gossip car il a un impact sur la réplication.

Le Gossip est une manière décentralisée pour un cluster d'avoir l'information sur ses noeuds. Le Gossip est très scalable et peut être utilisé sur des clusters composés de plusieurs centaines de noeuds. Avec un cluster aussi grand, l'utilisation des heartbeats trouve ses limites.

Le concept derrière Gossip est assez simple, et pourtant incroyablement efficace. L'idée est que les noeuds vont *parler* aux autres noeuds qui sont près d'eux dans le Hash Ring. Comme chaque noeud a la même importance, chaque noeud va *parler* à ses *voisins*.

Considérons l'exemple suivant :

- Le noeud c3 interroge le noeud c2 et lui demande son statut : tout va bien
- Le noeud c3 interroge le noeud c4 et lui demande son statut : problème apparent, le serveur ne répond pas
- Le noeud c1 interroge le noeud c3 et lui demande son statut : lui va bien, le noeud c2 aussi, par contre, le noeud c4 ne répond pas

Arrêtons là l'exemple, imaginons juste que ce processus est le même pour chaque noeud... Il n'est donc pas nécessaire d'utiliser une liste d'IPs, ni de heartbeats. Lorsqu'un nouveau noeud est ajouté à un cluster, en quelques minutes il aura la connaissance de tous les noeuds du cluster.

4.7 Etude pratique

4.7.1 Mise en place de l'environnement

Après une étude théorique de la réplication, il semble nécessaire de procéder à un expérimentation, afin de voir concrètement comment se comporte un cluster. Pour procéder, Docker sera largement utilisé. La mise en place de l'étude pratique comprend :

- La virtualisation de plusieurs noeuds Cassandra avec Docker
- Le paramétrage des noeuds pour les faire fonctionner en cluster
- La création d'un Keyspace, et l'insertion de données

Les commandes de mise en place de l'environnement composé de 5 noeuds fonctionnant en cluster, et de la création d'un keyspace *boredpanda*, et d'une table *data* se situent en annexe G. Les noeuds créés ont tous un token *CASSANDRA_TOKEN*, qui permet de :

- Placer les noeuds sur le Hash Ring
- Faire le lien avec le partitioner (Murmur3Partitioner par défaut) pour la répartition des ressources sur les différents noeuds

Le Keyspace boredpanda est créé avec un facteur de réplication de 3, et une stratégie de placement des réplicats simple. A ce stade, aucune donnée n'est insérée en base. L'état du cluster peut se visualiser ainsi :

```
$ sudo docker exec -it cassandra-1 /bin/bash
[docker]$ nodetool ring
Datacenter: datacenter1
```

```
==========
Address      Rack        Status State   Load            Owns                Token
                                                                            10000
172.17.0.9   rack1       Up     Normal  36.17 KB        100.00%             1
172.17.0.10  rack1       Up     Normal  45.43 KB        100.00%             10
172.17.0.11  rack1       Up     Normal  61.67 KB        0.00%               100
172.17.0.12  rack1       Up     Normal  45.53 KB        0.00%               1000
172.17.0.13  rack1       Up     Normal  77.92 KB        0.00%               10000
```

On a donc bien 1 cluster, dont le nom par défaut est *datacenter 1*, composé de 5 noeuds, tous activés, et organisés sur le Hash Ring selon leur token.

4.7.2 Insertion de données

Une donnée a été insérée en base. Les commandes d'insertion de la ligne sont situées en annexe G. Regardons maintenant où la donnée a été stockée. Pour cela, nous allons afficher les statistiques de chaque noeud, grâce à leur adresse IP. Les commandes sont en annexe G, et le résumé est le suivant :

```
$ sudo docker exec -it cassandra-1 /bin/bash
[docker]$ nodetool ring
Datacenter: datacenter1
==========
Address      Docs

172.17.0.9   1
172.17.0.10  1
172.17.0.11  1
172.17.0.12  0
172.17.0.13  0
```

Les données ont donc été répliquées sur 3 noeuds, la première écriture s'est faite sur le noeud 172.17.0.9 et sur les 2 noeuds suivants sur le Hash Ring. Le résultat est donc cohérent avec la stratégie simple de placements des réplicats.

4.7.3 Lecture

Connectons-nous sur le noeud 172.17.0.13. Ce noeud ne contient aucune ressource, et pourtant grâce au Hash Ring et à la réplication, Cassandra va s'en servir comme coordinateur, et va aller chercher la ressource là où elle est stockée.

```
$ sudo docker exec -it cassandra-5 /bin/bash
[docker]$ cqlsh
cqlsh > use boredpanda;
cqlsh:boredpanda > select * from data;

 id | value
----+----------------------
 10 | Here is a first data

(1 rows)
```

On notera que le keyspace a été créé sur le noeud 172.17.0.9. Pourtant le noeud 172.17.0.13 a la connaissance de la structure du keyspace. La requête de lecture renvoie bien la ressource alors qu'elle n'est pas stockée localement !

4.7.4 Cohérence des données en lecture

Pour étudier la cohérence des données en lecture, nous allons utiliser la ressource stockée, et stopper 2 noeuds Cassandra sur les 3. Pour ce faire, nous allons utiliser Docker

```
$ sudo docker pause cassandra-2
$ sudo docker pause cassandra-3
$ sudo docker exec -it cassandra-1 /bin/bash
[docker]$ nodetool ring
Datacenter: datacenter1
==========
Address      Rack        Status State   Load            Owns                Token
                                                                            10000
172.17.0.9   rack1       Up     Normal  55.88 KB        100.00%             1
172.17.0.10  rack1       Down   Normal  65.14 KB        0.00%               10
172.17.0.11  rack1       Down   Normal  68.54 KB        0.00%               100
172.17.0.12  rack1       Up     Normal  65.24 KB        0.00%               1000
172.17.0.13  rack1       Up     Normal  54.7 KB         0.00%               10000
```

Nous pouvons maintenant paramétrer le niveau de cohérence des données. Réalisons une requête de lecture. Le système est paramétré pour assurer la meilleure cohérence des données. On s'attend à ce que la requête plante car en mode ALL, Cassandra attend la réponse de tous les noeuds.

```
$ sudo docker exec -it cassandra-1 /bin/bash
[docker]$ cqlsh
cqlsh> use boredpanda;
cqlsh:boredpanda> consistency all;
Consistency level set to ALL.
cqlsh:boredpanda> select * from data;
Unable to complete request: one or more nodes were unavailable.
```

Comme attendu, la réponse renvoyée au client est une erreur. Testons maintenant le mode ONE, qui devrait normalement renvoyer la ressource du noeud le plus rapide. On s'attend à ce que la ressource du noeud 172.17.0.9 soit renvoyée.

```
$ sudo docker exec -it cassandra-1 /bin/bash
[docker]$ cqlsh
cqlsh> use boredpanda;
cqlsh:boredpanda> consistency one;
Consistency level set to ONE.
cqlsh:boredpanda> select * from data;

 id | value
----+----------------------
 10 | Here is a first data

(1 rows)
```

Dans ce schéma, le système est très disponible, mais ne vérifie pas la cohérence des données. Pour preuve, il renvoie effectivement la ressource au client alors que tous les autres noeuds qui contiennent la ressource sont perdus. Enfin, testons la stratégie du quorum. Avec 2 noeuds sur 3 perdus, la requête devrait normalement renvoyer au client une erreur.

```
$ sudo docker exec -it cassandra-1 /bin/bash
[docker]$ cqlsh
```

```
cqlsh> use boredpanda;
cqlsh:boredpanda> consistency quorum;
Consistency level set to QUORUM.
cqlsh:boredpanda> select * from data;
Unable to complete request: one or more nodes were unavailable.
```

Le résultat obtenu est bien celui attendu. Moins de la moitié des réplicats est disponible, la requête renvoie donc une erreur. Réactivons un noeud, et retestons.

```
$ sudo docker unpause cassandra-2
$ sudo docker exec -it cassandra-1 /bin/bash
[docker]$ nodetool ring
Datacenter: datacenter1
==========
Address      Rack        Status State   Load            Owns                Token
                                                                            10000
172.17.0.9   rack1       Up     Normal  55.88 KB        100.00%             1
172.17.0.10  rack1       Up     Normal  65.14 KB        0.00%               10
172.17.0.11  rack1       Down   Normal  68.54 KB        0.00%               100
172.17.0.12  rack1       Up     Normal  65.24 KB        0.00%               1000
172.17.0.13  rack1       Up     Normal  54.7 KB         0.00%               10000
[docker]$ cqlsh
cqlsh> use boredpanda;
cqlsh:boredpanda> consistency quorum;
cqlsh:boredpanda> select * from data;

 id | value
----+----------------------
 10 | Here is a first data

(1 rows)
```

Lorsque le noeud est réactivé (via Docker), il faut tout de même quelques dizaines de secondes avant qu'il soit effectivement réintégré dans le cluster. Le plus important est que la règle du quorum soit validée, avec 2 noeuds sur 3 disponibles, Cassandra accepte de retourner au client une ressource.

5 Discussion et développements futurs

5.1 Discussion sur l'approche de collecte de données

5.1.1 Forces

Une des forces de cette approche est d'avoir été capable d'extraire de l'information du site Internet boredpanda.com, sans jamais avoir à les solliciter. Le processus d'extraction des données a donc du faire preuve d'adaptation pour se conformer aux contraintes imposées comme le site, comme par exemple l'absence de flux RSS.

Le serveur NodeJS en fonctionnement est également autonome. Connecté à Cassandra, il est alors possible de le laisser agréger l'information des articles pendant plusieurs mois. Si le serveur venait à planter, un superviseur a spécialement été intégré pour le relancer aussitôt.

La récupération des données a été pensée pour se montrer discrète vis-à-vis des administrateurs du site boredpanda.com. Les articles sont postés sur le site à raison de environ 5 par jour. Sur l'ensemble d'une journée, le serveur NodeJS sera donc capable de récupérer le contenu de ces 5 articles ainsi que les informations de leurs auteurs en près de 10 requêtes, étalées sur la journée. Le risque d'alerter les administrateurs du site, et donc de faire bannir l'IP du service est donc très faible.

Le fait d'avoir choisi une base de données Cassandra offre la possibilité de poursuivre l'analyse de données avec des outils communément utilisés dans le big data, tels que Spark.

5.1.2 Faiblesses

Bien que l'architecture logicielle du projet soit robuste, elle repose sur l'extraction de données qui sont hébergées sur un site Internet externe, et c'est là la plus grande faiblesse de ce système. Le système peut tout à fait se trouver en difficulté, notamment face à des questions du type :

- Que faire si l'URL de la sitemap est modifiée, ou si boredpanda.com n'utilise plus de sitemap?
- Comment réagir si boredpanda.com décide de changer de template graphique (dans ce cas, toute la phase d'extraction de la donnée est à refaire)?
- Comment gérer le cas où boredpanda.com décide (comme beaucoup d'autres sites) de ne plus afficher le nombre de vues d'un article ?

Dans les deux premiers cas, il est toujours possible de mettre à jour l'algorithme d'extraction de l'information. Le risque pour que boredpanda.com n'utilise plus de sitemap est quant à lui quasi-nul, car l'utilisation des sitemaps est devenue un standard en SEO. Par contre, si boredpanda.com décide de ne plus afficher le nombre de vues de l'article, il sera alors impossible de collecter la variable métrique à expliquer. Si ce cas de figure devait se présenter, le projet s'arrêterait net, et il n'existe pas de solution de contournement.

5.2 Cassandra, base de données NoSQL

L'utilisation de Cassandra a fourni des résultats impressionnants. Les possibilités du schéma de stockage sont infinies, et permettent de traiter des sujets simples (comme c'est le cas dans ce projet) et de situations beaucoup plus complexes, avec beaucoup de fluidité.

Les essais de déploiement de Cassandra en mode cluster, avec réplication ont montré à quel point la distributivité chez Cassandra est puissante. Alors que chaque noeud joue son rôle de supervisor, la base de données se montre extrêmement robuste, notamment en cas de reprise sur panne.

Cassandra continue de monter en puissance dans l'écosystème des bases de données NoSQL. De plus en plus de déploiement de Cassandra sont réalisés dans le cadre d'applications BigData, si bien que en quelques années, Cassandra s'est avéré être LE choix NoSQL privilégié des architectures bigdata.

5.3 L'analyse prédictive de la popularité des articles

5.3.1 Nécessité de structurer les données

Les données extraites du site boredpanda ont été sauvegardées dans Cassandra. Cependant, dans le cadre d'une analyse prédictive, il sera nécessaire de structurer les données, car certaines informations pourraient tout à fait être redondantes et ainsi fausser les résultats. C'est le cas par exemple lorsque l'on stocke en même temps le nombre de mots dans le titre, et le ratio de stop-words du titre. De même, il faudra faire attention aux relations de colinéarité qui pourraient exister entre plusieurs variables. C'est le cas par exemple du nombre de vues et du nombre de partages sur Facebook.

Les données collectées ont été choisies selon un a priori, mais une analyse exploratoire sera nécessaire afin de filtrer l'information qui sera utilisée pour mener l'analyse de données.

Ce travail sur les variables est d'autant plus important que au rythme de 10 articles par jours en moyenne, on peut au mieux espérer lancer une analyse de données sur un jeu d'une centaine d'observations, après environ 1 mois de collecte. Avec moins de données, les erreurs liées aux traitement pré-analytiques ne seront pas noyés dans la masse. Si l'on voulait attendre d'avoir 1000 observations avant de lancer l'analyse, il faudrait alors patienter au moins 1 an!

5.3.2 Utilisation de Cassandra et Spark

Le choix de Cassandra s'est réalisé sur ses caractéristiques architecturales, et ses performances. Mais ce choix est également conditionné par la capacité du moteur de bases de données à retransmette efficacement l'information en vue d'une analyse de données dans un contexte de données massives. Et sur ce point, Cassandra brille également en utilisant des connecteurs qui lui permettent d'interagir directement avec Spark.

Dans le cadre d'une analyse de données massives, le workflow intégrerait donc naturellement Cassandra, Spark, et ses librairies de machine learning : MlLib si le langage utilisé est Scala, Scikit-learn si le langage utilisé est Python.

5.4 Développements futurs

5.4.1 Récupération d'autres catégories de variables explicatives

Même si les composantes d'un article peuvent expliquer sa popularité, on peut rester libre de penser que ce sont ici des conditions nécessaires mais non suffisantes pour pouvoir expliquer et prédire le nombre de vues d'un article par les internautes. La popularité d'un article se fait non seulement par son contenu, mais également par rapport à la popularité du thème traité. On peut donc imaginer que la popularité d'un article se fasse également sur la popularité des mots clés de l'article sur Internet.

Pour pousser plus loin l'extraction de contenu, il faudrait également disposer de métriques qui relatent le résultat d'une recherche des mots clés sur Google par exemple. Un moyen de récupérer cette information pourrait être de faire une recherche sur Google avec les mots clés de l'article, et considérer le nombre de documents que Google renvoie.

Sur le même principe, les informations liées à l'auteur de l'article ne sont pas suffisantes. Il faudrait pouvoir avec le même schéma de requête parser sa page Facebook pour en extraire par exemple son

nombre de *Followers*, le nombre d'articles qu'il a écrits, le nombre de *Likes*. Il serait ainsi possible d'extraire des variables périphériques à l'article, afin d'en avoir une vision encore plus ensembliste.

5.4.2 Outils de pérennisation de l'extraction de l'information dans les articles

Comme il a été discuté auparavant, une des faiblesses du système proposé est qu'il reste très tributaire du site boredpanda.com, et à la moindre modification sur le site, c'est tout le processus qui se bloque. Comme il n'est pas possible de s'assurer que le site boredpanda.com ne sera jamais mis à jour, une technique de protection peut être de mettre en place des systèmes de monitoring, qui auraient notamment la fonction suivante :

- Vérifier que toutes les variables explicatives sont renseignées
- Vérifier que le serveur NodeJS récupère bien environ 5 articles par jours
- Vérifier que l'URL des articles récupérée est bien valide

Si une de ces vérifications montre une erreur, alors un système d'alerte (un mail par exemple) pourrait prévenir du problème.

En plus de ce système d'alerte, une visualisation en temps réelle de l'activité du serveur pourrait être mise en place. Elle permettrait de faire le point sur le nombre d'articles récupérés, l'éventuelle latence dans le traitement des articles.

5.4.3 Généralisation du processus d'extraction sur d'autres sites

Le processus de collecte d'articles mis en place pour le site boredpanda.com pourrait tout à fait être utilisé pour n'importe quel autre site d'actualités. D'un site à l'autre, il faudra reprendre entièrement la phase d'extraction de données d'une page HTML, mais pour le reste, le schéma reste rigoureusement identique.

Néanmoins, pour que ce projet puisse être déployé sur un autre site d'actualité, quelques contraintes inhérentes au site sont à prendre en compte :

- Le site considéré doit montrer son code source en clair, non crypté comme ce peut être le cas sur certains sites
- Le site considéré doit donner accès aux variables métriques à expliquer, telles que le nombre de vues ou le nombre de partages sur les réseaux sociaux.
- Le site doit proposer une manière de récupérer la liste temps réel des articles postés, avec leur date de mise en ligne

6 Code source du projet

6.1 Code source du serveur NodeJS

Le code source du serveur NodeJS est disponible sur Git, à l'adresse suivante:

https://github.com/mysketches/boredpanda

Le repository contient le code Javascript. Le bon fonctionnement du serveur NodeJS requiert d'installer au préalable les dépendances, et une base de données Cassandra. Afin de permettre une intégration plus simple, le projet - dans son ensemble - a été encapsulé dans une image Docker.

6.2 Image Docker du projet

Pour ne pas avoir à installer sur un environnement de déploiement les packages système, tels que NodeJS, Cassandra et les dépendances, l'ensemble du système et de ses dépendances a été empaqueté dans une image Docker, disponible à l'adresse suivante :

https://hub.docker.com/r/mysketches/boredpanda/

L'image Docker se télécharge à l'aide de la commande suivante :

```
docker pull mysketches/boredpanda
```

Le lancement de l'image en tâche de fond se fait ainsi :

```
docker run -d -i -t --name boredpanda mysketches/boredpanda
```

A son lancement, la base de données Cassandra sera chargée, et le serveur NodeJS sera lancé. La récupération des articles dans le sitemap se fera par intervalle de 12 heures. Le démarrage de l'image rend l'ensemble du processus opérationnel.

L'image Docker a été déployée sur un serveur Web, et agrège actuellement les articles avec une cadence de environ 10-15 articles par jour. L'utilisation du mode daemon de Docker permet de faire tourner le serveur NodeJS en totale autonomie. Sur le même principe, d'autres projets de mise à disposition de services, hébergés sur le même serveur tournent en autonomie depuis maintenant plus de 7 semaines.

7 Bibliographie

Cassandra - The definitive guide
By Eben Hewitt
Publisher: O'Reilly Media
Ebook: November 2010

Cassandra High Availability
By Robbie Strickland
Publisher: Packt Publishing
Ebook: December 2014

Mastering Apache Cassandra, 2nd Edition
By Nishant Neeraj
Publisher: Packt Publishing
Ebook: March 2015

Learning Cassandra for Administrators
By Vijay Parthasarathy
Publisher: Packt Publishing
Ebook: November 2013

Berglund and McCullough on Mastering Cassandra for Architects
By Tim Berglund, Matthew McCullough
Publisher: O'Reilly Media
Video: January 2012

Appendices

A Présentation des variables liées au contenu textuel de l'article

is_article	Est-ce un article ou un formulaire
has_comments	Est-ce que l'article a des commentaires
title_words	Nombre de mots du titre
title_ratio	Ratio des non stop words dans le titre
content_words	Nombre de mots du contenu
content_ratio	Ratio des non stop words dans le contenu
description_words	Nombre de mots de la description
description_ratio	Ratio des non stop words dans la description
publish_date	Quel est le jour de la semaine de la publication
publish_weekend	Est-ce que l'article a été posté le week-end
h1	Nombre de balises H1
h2	Nombre de balises H2
h3	Nombre de balises H3
h4	Nombre de balises H4
more_info_facebook	Référence vers une page Facebook
more_info_instagram	Référence vers une page Instagram
more_info_imgur	Référence vers une page Imgur
more_info_flickr	Référence vers une page Fickr
more_info_etsy	Référence vers une page Etsy
more_info_site	Référence vers un site Web
more_info_tumblr	Référence vers une page Tumblr
more_info_twitter	Référence vers une page Twitter

B Présentation des variables liées aux catégories de l'article

tags	Nombre de tags
tags_length	Longueur des tags
category_travel	
category_photography	
category_animals	
category_illustration	
category_diy	
category_good_news	
category_funny	
category_art	
category_parenting	
category_other	
category_nature	
category_architecture	
category_product_design	
category_architecture	
category_style	
category_body_art	
category_digital_art	
category_painting	
category_food_art	
category_social_issues	
category_drawing	
category_sculpting	
category_entertainment	
category_advertising	
category_interior_design	
category_street_art	
category_video	
category_graphic_design	
category_weird	
category_optical_illusions	
category_history	
category_paper_art	
category_automotive	

category_technology

category_needle_and_thread

category_comics

category_packaging

category_science

category_recycling

category_typography

category_installation

category_land_art

category_furniture_design

category_pics

category_home

category_challenge

C Présentation des variables liées au contenu média de l'article

pictures	Nombre de photos
picture_horizontal	Nombre de photos horizontales
picture_vertical	Nombre de photos verticales
picture_square	Nombre de photos carrées
ratio_horizontal	Proportion de photos horizontales
ratio_vertical	Proportion de photos verticales
ratio_square	Proportion de photos carrées
videos	Nombre de vidéos

D Présentation des variables liées à l'auteur de l'article

author_posts	Nombre de posts
author_likes_per_post	Nombre de likes par post
author_views_total	Nombre total de vues
author_views_per_post	Nombre de vues par post
author_points_per_post	Nombre de points par post
author_featured_posts	Nombre de posts mis en avant
author_more_info_facebook	L'auteur a-t-il une page Facebook
author_more_info_instagram	L'auteur a-t-il une page Instagram
author_more_info_imgur	L'auteur a-t-il une page Imgur
author_more_info_flickr	L'auteur a-t-il une page Flickr
author_more_info_etsy	L'auteur a-t-il une page Etsy
author_more_info_site	L'auteur a-t-il un site Web
author_more_info_tumblr	L'auteur a-t-il une page Tumblr
author_more_info_twitter	L'auteur a-t-il une page Twitter

E Présentation des variables métriques à expliquer

views	Nombre de vues de l'article
share	Nombre de partages de l'article
vote	Nombre de votes des internautes

F Présentation de la structure de données dans Cassandra

```
create keyspace IF NOT EXISTS boredpanda with replication =
{'class':'SimpleStrategy', 'replication_factor':1};

USE boredpanda;

CREATE TABLE IF NOT EXISTS data (
   id text PRIMARY KEY,
   is_article boolean,
   metrics map<text, int>,
   comments map<text, boolean>,
   title map<text, float>,
   content map<text, float>,
   media map<text, float>,
   tags map<text, float>,
   description map<text, float>,
   date map<text, text>,
   category map<text, boolean>,
   h map<text, int>,
   more map<text, boolean>,
   author_metrics map<text, int>,
   author_links map<text, boolean>,
   author_info map<text, float>
);
```

G Mise en place de la réplication avec Cassandra

G.1 Création de l'environnement

```
$ sudo docker run -d -e "CASSANDRA_TOKEN=1"
    --name cassandra-1 spotify/cassandra:cluster
```

Nous avons besoin de l'IP du premier noeud pour créer les autres noeuds et faire fonctionner l'ensemble au sein d'un seul et même cluster

```
$ sudo docker inspect -f '{{.NetworkSettings.IPAddress}}' cassandra-1
127.0.0.9
```

```
$ sudo docker run -d -e "CASSANDRA_TOKEN=10" -e "CASSANDRA_SEEDS=172.17.0.9"
    --name cassandra-2 spotify/cassandra:cluster
$ sudo docker run -d -e "CASSANDRA_TOKEN=100" -e "CASSANDRA_SEEDS=172.17.0.9"
    --name cassandra-3 spotify/cassandra:cluster
$ sudo docker run -d -e "CASSANDRA_TOKEN=1000" -e "CASSANDRA_SEEDS=172.17.0.9"
    --name cassandra-4 spotify/cassandra:cluster
$ sudo docker run -d -e "CASSANDRA_TOKEN=10000" -e "CASSANDRA_SEEDS=172.17.0.9"
    --name cassandra-5 spotify/cassandra:cluster
```

G.2 Création d'un Keyspace, avec un facteur de réplication de 3

```
$ sudo docker exec -it cassandra-1 /bin/bash
[docker]$ cqlsh 172.17.0.9
cqlsh > CREATE keyspace boredpanda with replication = {'class':'SimpleStrategy',
    'replication_factor':3};

cqlsh > USE boredpanda;
cqlsh:boredpanda > CREATE TABLE data (
   id int,
   value text,
   PRIMARY KEY (id)
);
```

G.3 Ajout de données

```
$ sudo docker exec -it cassandra-1 /bin/bash
[docker]$ cqlsh 172.17.0.9
cqlsh > USE boredpanda;
cqlsh:boredpanda > INSERT INTO data (id, value) VALUES (10, 'Here is a first data');
```

G.4 Etat du cluster après insertion d'un document

```
$ sudo docker exec -it cassandra-1 /bin/bash
[docker]$ nodetool cfstats -h 172.17.0.9 boredpanda
Keyspace: boredpanda
   Read Count: 0
   Read Latency: NaN ms.
   Write Count: 1
   Write Latency: 0.637 ms.
[docker]$ nodetool cfstats -h 172.17.0.10 boredpanda
Keyspace: boredpanda
```

```
   Read Count: 0
   Read Latency: NaN ms.
   Write Count: 1
   Write Latency: 0.612 ms.
[docker]$ nodetool cfstats -h 172.17.0.11 boredpanda
Keyspace: boredpanda
   Read Count: 0
   Read Latency: NaN ms.
   Write Count: 1
   Write Latency: 0.474 ms.
[docker]$ nodetool cfstats -h 172.17.0.12 boredpanda
Keyspace: boredpanda
   Read Count: 0
   Read Latency: NaN ms.
   Write Count: 0
   Write Latency: NaN ms.
[docker]$ nodetool cfstats -h 172.17.0.13 boredpanda
Keyspace: boredpanda
   Read Count: 0
   Read Latency: NaN ms.
   Write Count: 0
   Write Latency: NaN ms.
```