Août 2015

Rapport de Stage

Acquisition de données textuelles liées à l'épidémiologie

Sous la tutelle de :

Mathieu ROCHE

Sylvain FALALA

Elena ARSEVSKA

Jocelyn DE GOER

BELOT Baptiste
IUT DE MONTPELLIER

Remerciements

Je souhaiterais remercier Mathieu Roche et Sylvain Falala pour nous avoir accueilli au sein du CIRAD pour effectuer ce stage et pour leur soutien.

Je voudrais remercier Madalina Croitoru, notre responsable à l'IUT de Montpellier pour l'enseignement qu'elle nous a apporté durant cette année spéciale en Informatique.

Je voudrais remercier Jocelyn De Goer, Informaticien et Doctorant à l'INRA au département LIMOS, pour son aide scientifique et technique continue.

Je souhaiterais remercier Elena Arsevska pour son aide apportée pour établir le contexte épidémiologique.

Je souhaiterais également remercier David Chavergnac pour son aide technique et le reste de l'équipe épidémiologique du CIRAD notamment Renaud Lancelot et Andrea Apolloni pour leur professionnalisme et leur sympathie.

Pour finir, je souhaiterais spécialement remercier Clément Hemeury pour avoir été un bon partenaire lors de ce stage et pour son amitié tout au long de cette année scolaire.

Sommaire

Introduction

Epidémiologie et maladies infectieuse émergentes

L'épidémiologie est l'étude de la distribution et des déterminants des états ou des événements (y compris les maladies) liés à la santé, et de l'application de cette étude à la lutte contre les maladies et autres problèmes de santé. Les principes de l'épidémiologie de la santé animale sont identiques à ceux de l'épidémiologie de la santé humaine, à l'exception qu'ils appliquent aux populations animales. En tant que tel, l'épidémiologie de la santé humaine et animale fait parties de la même discipline (WHO, 2014).

L'un des rôles clés de l'épidémiologie est d'identifier et d'évaluer les menaces de maladies infectieuses émergentes. Ces menaces peuvent provenir de maladies exotiques ou d'agents pathogènes inconnus, avec un impact potentiellement négatif sur la santé, l'économie des pays touchés et le commerce international (Peeler & Taylor, 2011).

Au cours des dernières années, un certain nombre de maladies infectieuses émergentes se sont rapidement propagées au niveau international, provoquant de graves répercussions sur la santé animale.

En 2007, une maladie animale infectieuse émergente, la peste porcine africaine (PPA), réapparait après 30 ans d'absence en Europe de l'Est. La peste porcine africaine est une maladie hémorragique

hautement contagieuse et mortelle qui contamine les porcins domestiques et sauvages. En raison de l'importance de cette contamination, l'Organisation Mondiale de la Santé Animale (OIE) se doit de veiller sur l'évolution de la peste porcine africaine (OIE, 2015). Des enquêtes de terrain suggèrent que le transport de produits de viande infectée par les navires internationaux, destiné à nourrir les porcs locaux autour du port de Sotchi en Géorgie, est à l'origine de l'émergence de la PPA. En outre, les services vétérinaires ont mis deux mois pour confirmer et notifier officiellement qu'il s'agissait bien de la peste porcine africaine, tandis que le virus s'était déjà répandu dans plusieurs pays d'Europe Orientale. La PPA représente aujourd'hui une grande menace pour le reste du continent européen (Sánchez-Vizcaíno, Mur, et Martínez-López, 2013).

La Veille Sanitaire Internationale

Dans un environnement en rapide évolution, il devient difficile pour les institutions nationales chargées de la sécurité sanitaire de se fier exclusivement aux systèmes traditionnels de surveillance des maladies qui n'ont pas été conçus pour identifier l'émergence de risques nouveaux. La Veille Sanitaire Internationale (VSI) fournit un cadre conceptuel qui permettra aux pays d'adapter leurs systèmes de surveillance national pour répondre à ce nouveau défi.

La VSI inclut toutes les activités liées à l'identification précoce des risques potentiels pour la santé, leur vérification et leur évaluation, et l'investigation permettant de faire des recommandations de mesures de contrôle en santé publique. La VSI intègre deux types de composante, l'une basée sur des évènements et l'autre sur des indicateurs. Cette dernière composante correspond aux données structurées recueillies via les systèmes de surveillance en routine. Quant à la composante 'évènementielle' elle se rapporte aux données non structurées captées à partir de sources d'informations de toute nature (Paquet et al. 2006).

La VSI est attentive aussi bien sur les premiers signes d'épidémie mais est aussi vigilante sur les cas de contamination avancée et trace constamment l'évolution de la maladie. Son objectif est de rendre la détection de potentielles menaces pour la santé plus rapide et aisée, et de permettre un temps de réponse opportun. La spontanéité du web et des medias électronique en fait les principales sources d'information de la VSI.

Ce stage a donc été réalisé dans l'idée de veiller sur ces médias, et principalement le web, qui peuvent potentiellement regorger d'informations concernant les dangers sanitaires pour les populations animales.

Actuellement, il existe plusieurs systèmes de veille au niveau international basés sur le suivi de danger en santé humaine et/ou animale grâce à des informations provenant de médias.

Des systèmes automatisés tel que Medisys, HealthMap et Biocaster recueillent des articles provenant du web avec un intérêt potentiellement pertinent pour la santé publique, cela en temps réel et avec presque aucune intervention humaine. Ces systèmes publient des statistiques, afin de suivre l'évolution de l'épidémie sur des graphiques ou encore des cartes. Ces systèmes se basent principalement sur des méthodes d'extraction de données dans des textes qui reconnaissent certain termes ou combinaisons de termes spécifiques.

D'autres systèmes sont plus manuels et nécessitent une intervention humaine plus grande (à des degrés divers). Ces systèmes sont composés d'analystes formés qui évalue le contenu d'articles qui ont été choisi par pré-filtrage automatique. RMISP, ProMed et Argus sont des exemples de systèmes

à modération humaine bien établies dans le monde de la surveillance de la santé.

L'objectif de ce stage est donc de créer un outil permettant la détection automatique d'informations provenant de dépêches publiées sur le web à partir de mots clés pertinents sur la santé animale (maladies, symptômes...). Cet outil, nommé outil de Web Scraping, doit pouvoir sélectionner les articles qui peuvent renseigner sur des cas de maladies connues mais aussi de maladies infectieuses émergentes.

Le web étant une source extrêmement large et variée, nous avons décidé de nous concentrer sur la veille basée sur la surveillance de flux RSS.

Nous avons choisi d'utiliser les flux RSS car ils constituent une source riche et simple à exploiter. Mis à jour automatiquement et régulièrement, ils constituent une source potentiellement pertinente vis-à-vis de la veille de maladies émergentes animales. De plus, les systèmes de veille déjà existant comme HealthMap et Biocaster se servent principalement des flux RSS comme source d'informations pour les maladies humaines.

Cahier des Charges

Nous allons dans ce chapitre aborder les différentes spécifications de notre outil de Web Scraping développé dans le cadre de ce stage. Cet outil fait partie du processus d'un système de veille sanitaire. Les textes qui en ressortent sont exploités par un autre outil développé par mon camarade Clément Hemeury dans le cadre de son stage pour l'année spéciale Informatique à l'IUT de Montpellier.

Ce système permettra d'aspirer des articles web provenant de sources pertinentes, d'en nettoyer le texte de tout élément non désiré (balises HTML, publicité...) et d'appliquer des règles grammaticales pour déterminer le sens des phrases afin de récupérer les informations utiles.

Dans ce document, nous allons seulement nous attarder sur l'aspiration des articles et le nettoyage du texte.

Aspiration d'articles

INPUT

Une liste d'adresses de flux RSS provenant de sources jugées pertinentes. Ces adresses sont stockées dans une base de données et de nouveaux flux peuvent être ajoutés sans perturber le script.

OUTPUT

Le titre, la description, et l'URL d'un article web aspiré du flux RSS sans filtrage ; à partir du moment

où l'article figure dans le flux RSS, ses différentes données qui sont dans le flux RSS (titre, lien et description) seront aspirés.

Description de la fonction

Cette fonction va nous permettre de récupérer l'article tel qu'il est représenté dans le flux RSS, c'est à dire, avec son titre, son lien et sa description.

Suite à cela, le titre de l'article et la description vont nous permettre de réaliser une étape de pré-filtrage afin d'éliminer les articles hors-sujets. Le titre servira également lorsque les articles seront enregistrés et classés dans la base de données. De plus, le titre sera haché afin d'attribuer un identifiant à chaque article. Ces étapes ne font pas parties de la fonction d'aspiration.

Les flux RSS, aussi appelés flux XML, contenu syndiqué ou flux Web, contiennent des informations fréquemment mises à jour et publiées par un site Web. Les flux sont souvent utilisés par les sites Web d'actualités et les blogs, mais ils peuvent aussi servir à distribuer d'autres types de contenu numérique, notamment des images et des fichiers audio ou vidéo (http://windows.microsoft.com/fr-fr/windows7/what-is-a-feed-rss). Ce sont en fait des fichiers texte formaté en XML afin d'être lu par des lecteurs de flux. Les fichiers sont formatés différemment selon le contenu.

Par exemple, si un site dispose d'un flux RSS et publie un nouvel article, il apparaîtra sur le flux avec son titre, la date, l'heure, une brève description et l'URL menant à la page web de l'article (en lien hypertexte sur le titre).

fig 1: Flux RSS de HealthMap (http://www.healthmap.org/promed2.xml) avec le premier article (visionné sur un navigateur).

Il faudra donc récupérer les 3 éléments qui nous intéresse, soit le titre, le lien et la description de chaque article du flux.

Nettoyage du code HTML et récupération du texte

INPUT

L'URL qui mène à la page web contenant l'article.

OUTPUT

Une chaine de caractères correspondant au texte de l'article, nettoyé de toutes balises HTML, scripts ou style.

Description de la fonction

Le but de cette fonction sera d'extraire le texte de l'article du code source HTML de la page web.

Cette fonction doit permettre de supprimer toutes balises, scripts, style et en-tête du code source et de détecter le texte de l'article pour l'extraire et le retourner. Suite à cette fonction, le texte sera

enregistré sur la base de données dans une table qui correspond aux articles web qui n'ont pas été triés et classés comme articles pertinents (opération réalisée par l'outil en aval).

Dû à la trop grande disparité du code source entre les pages web et de l'absence de schéma bien précis dans ces dernières, la fonction ne pourra pas être à 100 % fiable. Une évaluation sera donc effectuée afin de déterminer une estimation de l'efficacité de la fonction.

Pré-filtrage des articles et méthodes de sélection

Avant de nettoyer les textes et de les enregistrer, nous devons réaliser une épate de pré-filtrage afin d'éliminer certains articles totalement hors-sujet. Pour ce faire, nous allons nous baser sur un dictionnaire de nom de maladies exotiques afin d'effectuer un premier filtre. Ce filtre sera appliqué à la fois sur le titre de l'article mais aussi sur sa description fournie par le flux RSS.

Il consistera à utiliser des expressions régulières pour déterminer si un mot clé ou une combinaison de mots clés est présent dans le titre et la description de l'article.

Les Expressions Régulières

Les expressions régulières, aussi appelées expressions rationnelles, sont une écriture compacte pour représenter un ensemble (peut-être infini) de séquences de caractères semblables. Représentées en chaines de caractères, elles permettent de décrire un ensemble variable par l'utilisation d'une syntaxe précise, qui se retrouve dans une foule de langages et d'outils.
(http://edutechwiki.unige.ch/fr/Expression_régulière)

Exemple d'une expression régulière :

(\W|^)swine\flu(\W|$)

Cette expression permet de rechercher une correspondance avec l'expression « swine flu ».

- \W désigne tout caractère qui n'est pas une lettre, un chiffre ou un trait de soulignement. Ce méta caractère évite que l'expression régulière ne prenne en compte des caractères figurant avant ou après la proposition.
- ^ désigne le début d'une nouvelle ligne. Ce méta caractère indique à l'expression régulière de renvoyer la proposition si elle apparaît au début d'une ligne, c'est-à-dire sans caractère devant.
- $ désigne la fin d'une ligne. Ce méta caractère indique à l'expression régulière de renvoyer la proposition si elle apparaît à la fin d'une ligne, c'est-à-dire sans caractère derrière.

Dans notre cas, « swine flu » fait partie des mots clés de notre dictionnaire. Il faudra donc chercher une correspondance dans le titre et la description de l'article que l'on veut garder ou non.

Si jamais l'article ne contient pas de nom de maladies, alors un second test sera appliqué avec la même méthode, mais avec un dictionnaire de symptômes viraux. Cela permettra de détecter les articles mentionnant des maladies non identifiées, articles qui peuvent potentiellement soulever l'émergence d'une maladie infectieuse.

Rapport technique

Choix technologiques

Notre outil se base sur LAMP : Linux + Apache + MySQL + PHP

L'outil de Web Scraping sera exécuté quotidiennement sur un serveur dédié du CIRAD. Les articles retenus y seront également stockés.

- Nous avons choisi le PHP comme langage de programmation car, étant le langage web le plus répandu, de nombreuses sources sur le Web Scraping et le nettoyage de page web existent. De plus l'exploitation de base de données est extrêmement simple. PHP supporte aussi de nombreux protocoles et notamment le protocole HTTP qui nous permettra de récupérer les pages web.

- Nous avons utilisé phpMyAdmin avec MySQL comme outil de gestion de base de données pour son interface simple et son utilisation intuitive nous permettant de visualiser les tables aisément.

- Durant le développement, un serveur LAMP sur une machine locale a été utilisé afin de disposer de PHP, Apache et MySQL.

- Les scripts PHP ont été développé sous Geany. Ce logiciel avait été utilisé durant notre formation. Afin d'assurer une maintenance facile du notre application, les scripts PHP ont été développé en suivant le concept d'objet et de classe.

- CURL a été utilisé pour récupérer le code source des pages web. Il permet de faire passer notre script pour un navigateur et utilisateur normal, afin d'éviter les bloqueurs de robot.

- Nous avons utilisé les tâches planifiées Cron afin d'exécuter le script d'aspiration 2 fois par jour.

Ces technologies ont été choisies car nous les avons, pour la plupart, étudiées au cours de notre formation d'année spéciale Informatique. Le temps d'apprentissage étant alors moins long, il a fallu beaucoup moins d'adaptation vis à vis de ces technologies tout en étendant notre connaissance sur les langages utilisés, le Web Scraping étant une discipline que nous n'avons jamais abordé en cours. L'installation d'un serveur Apache sur une machine locale a aussi été une première.

Conception

En résumé, l'outil de Web Scraping utilisera donc :

- Une base de données afin d'enregistrer les articles et de stocker la liste des sources permettant de récupérer ses articles (Ex : liste de flux RSS...)

- Un script d'aspiration et de nettoyage d'article : En utilisant CURL, nous pourrons récupérer le code source des pages web provenant des différentes sources. La librairie ReabAbility (PHP) nous permettra de nettoyer la page web de toute balise HTML et de ne récupérer que l'article de la page.

- Un serveur Linux disposant de PHP et MySQL pour assurer les deux premiers points.

Base de données

Une base de données MySQL a été mise en place sur le serveur dédié afin de stocker les différentes données utiles à l'aspiration d'article (dictionnaire de mots clés, liste d'URL de flux RSS...) mais aussi pour stocker les résultats du script (articles aspirés avec leur titre et leur adresse).

Nous avons donc établi 3 tables afin de répondre à nos besoins

Modèle Conceptuel de Données (MCD) :

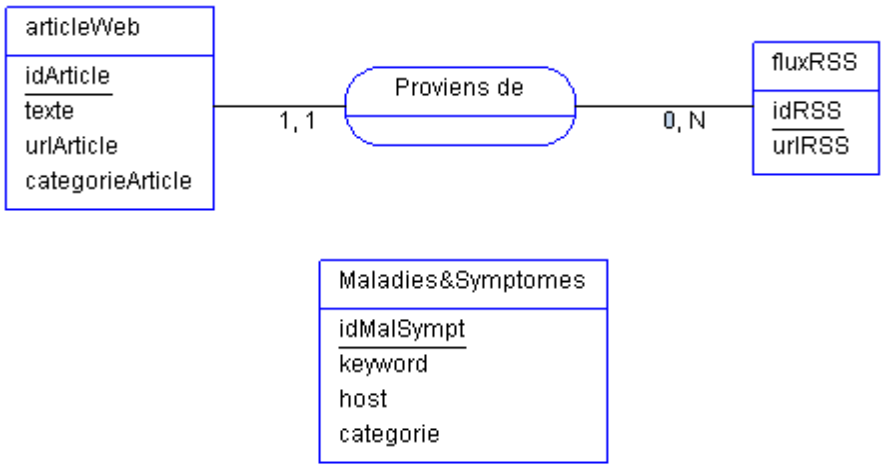


fig 2: Modèle Conceptuel de Données de notre base de données

Le modèle de la figure 2 (*fig 2)* présente une association entre articleWeb et fluxRSS.
Les cardinalités nous précisent que pour un article donné, un flux RSS lui est associé qui est celui dont l'article provient.

A partir de ce schéma, nous pouvons donc en déduire le Modèle Logique de Données (MLD).

Modèle Logique de Données (MLD) :

fluxRSS (idRSS, urlRSS)
Maladies&Symptomes (idMalSympt, keyword, host, categorie)
articleWeb (idArticle, texte, urlArticle, categorieArticle, #idRSS)

On obverse l'association entre fluxRSS et articleWeb représenté par la clé étrangère dans la table articleWeb.

Maladies & Symptomes :

Cette table constitue notre dictionnaire de mots clés pour les maladies et les symptômes. Les maladies sont rangées avec l'id le plus bas pour que le script puisse parcourir d'abord les mots clés concernant les maladies et ensuite les symptômes. Une catégorie est attribuée à chaque maladie et symptôme afin de pouvoir classer les articles jugés pertinents. D'autant plus qu'une multitude d'expressions et d'écritures existe pour une seule maladie.

La catégorie des maladies correspond à leur nom de base alors que les symptômes sont classés par syndrome.

NOM DE LA MALADIE	CATEGORIE
African Horse Sickness	African Horse Sickness
Equine Plague	African Horse Sickness
African swine fever	African swine fever
Wart-Hog Disease	African swine fever
Wart Hog Disease	African swine fever
Asfivirus infection	African swine fever
Bluetongue	Bluetongue
Sheep catarrhal fever	Bluetongue
Ovine catarrhal fever	Bluetongue
Bovine brucellosis	Bovine brucellosis
B.abortus	Bovine brucellosis
Brucella abortus	Bovine brucellosis
Bang's Disease	Bovine brucellosis
Bang Disease	Bovine brucellosis

fig 3 : Tableau d'exemples de mots clés pour les maladies

NOM DU SYMPTOME	CATEGORIE
livestock deaths	General
general clinical signs	General
weakness	General
mortality	General
fever	General
embryonic death	Reproductive
reproductive disorders	Reproductive
abortion	Reproductive
bucal ulcer	Mucus/cutaneous
nasal discharge	Mucus/cutaneous
gingival ulcers	Mucus/cutaneous
hyper salivation	Mucus/cutaneous
bucal lesions	Mucus/cutaneous
teat lesions	Mucus/cutaneous

fig 4 : Tableau d'exemples de mots clés pour les symptômes

Note : Les mots clés ont été sélectionnés selon des critères épidémiologiques par Elena ARSEVSKA, épidémiologiste au CIRAD et tutrice de ce stage.

Classe Flux RSS

Objectif

Cette classe va nous permettre de recevoir un fichier XML provenant d'un flux RSS grâce à une URL afin d'en extraire les informations des articles publiés chaque jour. Cette classe doit aussi nous permettre de créer des flux RSS provenant de Google News à partir de mots clés (Voir fonction creer_rss_gnews).

Constructeur et attribut

Le constructeur de la classe reçoit une URL afin de construire un objet flux RSS. Les URL des flux sont récupérés depuis une base de données et correspondent à des sources qui peuvent être potentiellement pertinentes (Par exemple : http://www.healthmap.org/promed2.xml qui provient de ProMED)

Attributs :

URL : adresse du flux RSS

arrayArticles : tableau d'objet articleRss qui contient les informations des articles (titre + description + lien) récupérés à l'adresse de l'URL

Fonction chargementRSS

Objectif

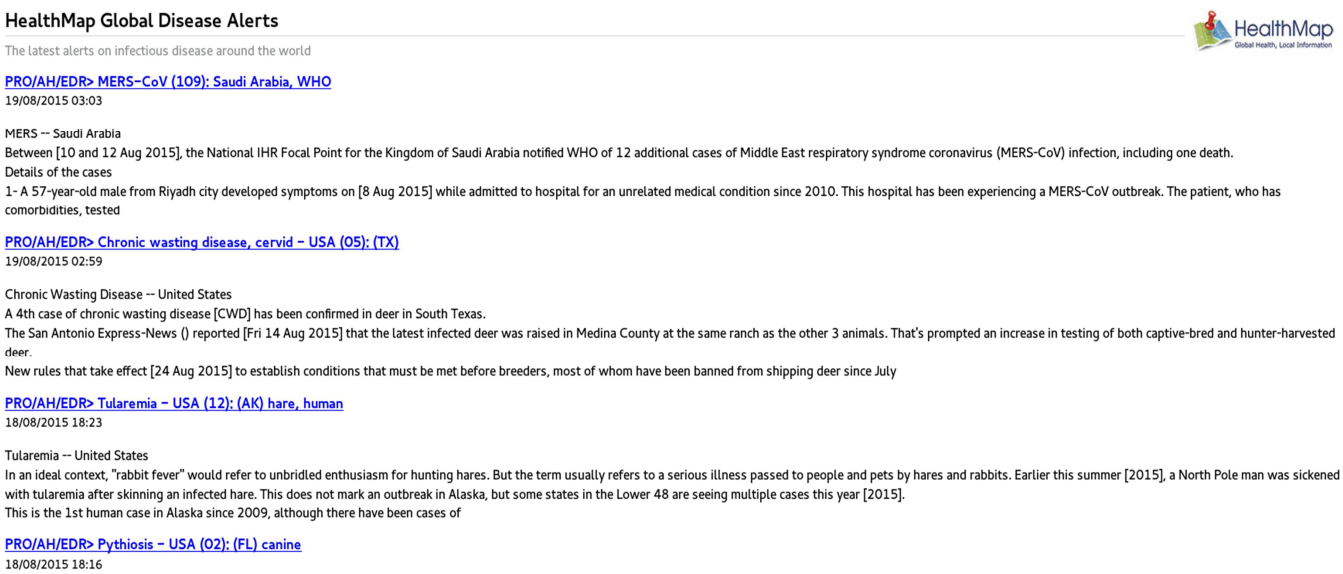
La fonction va nous permettre d'extraire le titre, la description et le lien des articles à l'URL du flux RSS et de les stocker dans arrayArticles sous forme d'objet de type *articleRss* avec le titre de l'article récupéré, le lien vers l'article et sa description donnée par le flux.

Input/Output

Entrée : On se sert de l'URL pour accéder au flux.

Sortie : On récupère tout les articles (ou *item* du flux) que l'on stocke dans l'attribut *arrayArticles*

Chaque flux RSS publie chaque jour plusieurs articles provenant d'un ou plusieurs sites. Ce flux peut aussi être accessible par navigateur.


HealthMap Global Disease Alerts

The latest alerts on infectious disease around the world

PRO/AH/EDR> MERS-CoV (109): Saudi Arabia, WHO
19/08/2015 03:03

MERS -- Saudi Arabia
Between [10 and 12 Aug 2015], the National IHR Focal Point for the Kingdom of Saudi Arabia notified WHO of 12 additional cases of Middle East respiratory syndrome coronavirus (MERS-CoV) infection, including one death.
Details of the cases
1- A 57-year-old male from Riyadh city developed symptoms on [8 Aug 2015] while admitted to hospital for an unrelated medical condition since 2010. This hospital has been experiencing a MERS-CoV outbreak. The patient, who has comorbidities, tested

PRO/AH/EDR> Chronic wasting disease, cervid - USA (05): (TX)
19/08/2015 02:59

Chronic Wasting Disease -- United States
A 4th case of chronic wasting disease [CWD] has been confirmed in deer in South Texas.
The San Antonio Express-News () reported [Fri 14 Aug 2015] that the latest infected deer was raised in Medina County at the same ranch as the other 3 animals. That's prompted an increase in testing of both captive-bred and hunter-harvested deer.
New rules that take effect [24 Aug 2015] to establish conditions that must be met before breeders, most of whom have been banned from shipping deer since July

PRO/AH/EDR> Tularemia - USA (12): (AK) hare, human
18/08/2015 18:23

Tularemia -- United States
In an ideal context, "rabbit fever" would refer to unbridled enthusiasm for hunting hares. But the term usually refers to a serious illness passed to people and pets by hares and rabbits. Earlier this summer [2015], a North Pole man was sickened with tularemia after skinning an infected hare. This does not mark an outbreak in Alaska, but some states in the Lower 48 are seeing multiple cases this year [2015].
This is the 1st human case in Alaska since 2009, although there have been cases of

PRO/AH/EDR> Pythiosis - USA (02): (FL) canine
18/08/2015 18:16

fig. 5 : exemple de flux RSS provenant de HealthMap sur navigateur avec tous ses articles

Etapes de la fonction

Un flux RSS est en fait un fichier texte sous un format XML. Sous PHP, la fonction *simplexml_load_file($URL)* nous permet grâce à une requête HTTP de récupérer ce fichier XML à l'URL passée en paramètre. Cette fonction retourne un objet de la classe *SimpleXMLElement* dont les propriétés contiennent les données du document XML récupéré.

Un item du flux RSS nous donne le titre de article ainsi que le lien, la date de publication et une brève description, sous la balise <channel> soit l'attribut du même nom dans l'objet *SimpleXMLElement* récupéré avec *simplexml_load_file($URL).*

```
$flux = simplexml_load_file($this->URL) ;
```

```
$donnée = $flux->channel ;
```

Le channel comporte alors une section *item* où figure les attributs de chaque article publiés (titre, lien, description...etc.).

La structure globale d'un flux RSS en XML ressemble donc à ceci :

```
<channel>
        <item1>
                        <title> titre </titre>
                        <link> lien </link>
                        <desc> description </desc>
                        ...
        </item1>
        <item2>
                        ...
        </item2>
                        ...
</channel>
```

Chaque balise correspond donc à un attribut de l'objet de type *SimpleXMLElement.*

Le programme va donc parcourir chaque item du flux et enregistrer le titre, le lien et la description dans un objet de type *articleRss* (Voir ...) et stocker cette article dans le tableau *arrayArticles.*

```
foreach($donnee->item as $valeur)
{
```

```
        $this->arrayArticles[]= new articlesRss($valeur->title,$valeur->link,$valeur->description);
}
```

Fonction creer_rss_gnews

Objectif

Le but de cette fonction est de pouvoir accéder à des flux RSS provenant de Google News en spécifiant un ou plusieurs mots clés. Les flux Google News permettent en effet d'accéder à des articles parus dernièrement grâce à des mots clés, permettant une recherche bien plus précise.

Ces mots clés sont spécifiés dans l'URL du flux RSS Google News.

Exemple avec les mots clés « Blue tongue » :

*https://news.google.com/news/feeds?pz=1&cf=all&ned=en&q=**Blue**+**tongue**&output=**rss***

Grâce à cette fonction, nous allons donc pouvoir créer des sources RSS à partir d'une liste de mots clés pertinentes

Input / Output

Entrée : Une chaine de caractère contenant un ou plusieurs mots clés. S'il y a plus d'un mot clé, ils doivent être séparés par un espace ou un « + » pour correspondre avec la syntaxe des URLs Google.

Sortie : Une chaine de caractère correspondant à l'URL formaté donnant accès au flux RSS ciblant les articles contenant les mots clés.

Etapes de la fonction

Il faut dans un premier temps, formater les mots clés pour les faire correspondre à la syntaxe de Google. En effet, dans les URLs des flux Google News, les différents mots clés sont séparés par des caractères « + ». Il suffit donc de remplacer les espaces de notre chaine de caractères par des « + ».

```
$mot_clés_google = str_replace(" ", "+" ,$mot_clés);
```

Il ne reste plus qu'à placer la chaine dans l'URL et de retourner le résultat :

```
$urlRSS =
"https://news.google.com/news/feeds?pz=1&cf=all&ned=en&q=".$mot_clés_google."&output=rss";

return $urlRSS ;
```

Classe articleRss

Objectif

articleRss va représenter les articles aspirés depuis les flux Rss. La classe nous permettra de stocker mais aussi de manipuler le titre, le lien et la description afin de pouvoir accéder a la page web du site de l'article.

Constructeur et attribut

Le constructeur de la classe reçoit 3 paramètres qui sont :

- le titre de l'article ($titre)
- le lien vers la page web de l'article ($lien)
- la description de l'article ($description)

Ces trois paramètres constituent les attributs de notre classe et sont donnés par le flux RSS lors de la consultation de ce dernier.

Cette classe ne contient pas de fonction particulière mais seulement des accesseurs (getLien, getDescription, getTitre) qui donne accès aux valeurs des attributs tout en les gardant privé.

Classe pageWeb

Objectif

Cette classe va nous permettre de décrire une page web (ou plus précisément, un article web) grâce à son URL, le code source de la page et le texte de l'article qui se trouvent sur la page. Elle devra être capable de récupérer le code source de la page lorsque l'URL est passée dans le constructeur. Elle disposera aussi d'une fonction qui « nettoiera » le code source de la page pour ne garder seulement que le texte de l'article de la page.

Constructeur et attribut

En paramètre, **peut** recevoir l'URL d'une page web. Si une URL est passée en paramètre, alors le constructeur appelle la fonction obtenirSource() qui permet de récupérer le code source de la page web à l'adresse donnée.

Attributs :

URL : adresse web de la page qui contient l'article.

source : code source de la page.

articleNettoyé : chaine de caractère qui contient le texte de l'article suite au nettoyage de la page.

Fonction obtenirSource

Objectif

Cette fonction va utiliser la bibliothèque libcurl afin de récupérer le code source d'une page web. Afin de contourner les bloqueurs de crawleurs de certains sites web, nous devrons configurer CURL pour faire passer notre script pour un « User Agent » utilisant Firefox.

Input / Output

Entrée : CURL doit recevoir une URL afin d'accéder à la page web. On lui passe donc l'adresse en attribut de la page à récupérer.

Sortie : CURL nous retourne alors la source de la page, c'est-à-dire, le code HTML de la page.

CURL et la libraire libcurl

CURL est un outil en ligne de commande open source qui permet de transférer des données grâce à une syntaxe URL (http://curl.haxx.se/). PHP se sert de libcurl qui est une bibliothèque créée par Daniel Stenberg et qui permet de se connecter et de communiquer avec différents types de serveurs, et ce, avec différents types de protocoles. libcurl supporte actuellement les protocoles HTTP, HTTPS, FTP, Gopher, Telnet, DICT, File et LDAP (http://php.net/manual/fr/intro.CURL.php).

Nous allons nous servir du protocole HTTP afin d'accéder à la page web.

Etapes de la fonction

Il faut dans un premier temps initialiser une session CURL grâce à la commande *CURL_init()* pour ensuite appliquer une gamme d'option afin de se faire passer pour un navigateur.

```
$CURL = CURL_init();

CURL_setopt_array($CURL, array(
                        CURLOPT_FOLLOWLOCATION => true,
                        CURLOPT_HEADER  => false,
                        CURLOPT_HTTPGET => true,
                        CURLOPT_RETURNTRANSFER => true,
                        CURLOPT_TIMEOUT => 10000,
                        CURLOPT_URL => '' $url '',
                        CURLOPT_USERAGENT => 'Mozilla/5.0 (Windows; U; Windows NT 5.1;
en-US; rv:1.8.1.13) Gecko/20080311 Firefox/2.0.0.13'
                    ));
```

CURLOPT_FOLLOWLOCATION => true : Certaine URL peuvent nous faire rediriger vers une autre adresse lorsque que l'on exécute une requête sur cette première. Cette option permet à CURL de suivre la nouvelle URL.

CURLOPT_HEADER => false : FALSE pour ne pas inclure l'en-tête dans la valeur de retour.

CURLOPT_HTTPGET => true : Permet d'initialiser la méthode de requête HTTP à GET.

CURLOPT_RETURNTRANSFER => true : Retourne directement le transfert sous forme de chaîne de la valeur retournée par *CURL_exec()* au lieu de l'afficher directement.

CURLOPT_TIMEOUT => 10000 : Règle sur 10 secondes le temps d'exécution de la fonction CURL.

CURLOPT_URL => '' $url '' : URL à récupérer.

CURLOPT_USERAGENT =>'Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.13) Gecko/20080311 Firefox/2.0.0.13' : Le contenu de l'en-tête *"User-Agent: "* à utiliser dans la requête HTTP. Ici, on précise que l'user agent utilise Mozilla Firefox sous Windows en anglais (US).

Source : http://php.net/manual/fr/function.CURL-setopt.php

On exécute ensuite la requête HTTP via CURL et on ferme la session :

```
$output = CURL_exec($CURL);
CURL_close($CURL);
```

La fonction nettoyerPage et Readability

Objectif

En utilisant la librairie Readability pour PHP, l'objectif de cette fonction sera de nettoyer le contenu d'une page web (balises, script...) afin de ne récupérer que le texte de l'article qui nous intéresse.

Input / Output

Entrée : Cette fonction reçoit en entrée le code source de la page web que l'on veut nettoyer. Les balises HTML, les styles et les éventuels scripts sont toujours présents dans le code.

Sortie : Retourne le texte de l'article présent sur la page web s'il a été détecté. La fonction ne garantit pas à 100 % que l'article soit correctement extrait. Une évaluation sera donc réalisée afin de déterminer son efficacité.

La librairie php-Readability

Readability a d'abord été une libraire JavaScript écrite par Arc90 afin de pouvoir formater des documents texte sur le web et d'extraire de l'information (Text Mining et Web Parsing).

L'utilisateur « feelinglucky » de Github a alors développé un portage pour PHP version 5. php-Readability a donc été écrit avec DOM (Document Object Model http://www.php.net/manual/en/book.dom.php).

Etapes de la fonction

Il faut d'abord créer un objet de type Readability et lui attribuer le code source de la page avec son URL :

```
$readability = new Readability($html, $this->URL);
$result = $readability->init();
```

La fonction *init()* permet entre autres de formater le texte de la page dans les différents attributs de la classe Readability comme le titre de l'article et son contenu récupérable par des accesseurs :

```
$this->articleNettoyé= $str.$readability->getContent()->textContent."";
```

Cette fonction retourne TRUE si l'exécution s'est bien faite, et FALSE dans le cas contraire. Il convient donc de tester le bon fonctionnement du code :

```
if ($result) {
        $str= '"' ;
        $str = $str.$readability->getContent()->textContent. '"';
        }
else{
        $str = 'Contenu Introuvable.' ;
}
$this->articleNettoyé = $str ;
return $str ;
```

articleNettoyé contiendra donc l'article même de la page web, débarrassé de toute balise, styles, où script que le code source pouvait contenir ou 'Contenu Introuvable ' si jamais l'extraction de l'article a échoué.

Déroulement du script « main.php »

L'utilisation de ces classes sera centralisée dans un script « main » qui se chargera d'appeler les constructeurs des différentes classes, et de mettre en œuvre les différentes fonctions.

L'acquisition de données textuelles suit une procédure bien précise. La figure 3 (*fig 3)* représente un schéma de l'architecture générale et du déroulement de la procédure.

Sur la figure 3:

(1) : La liste d'URL des flux RSS est récupérée depuis la base de données. Pour chaque URL récupéré, on construit un nouvel l'objet fluxRSS et on le stocke dans un tableau.

```
//$res est un tableau contentant la liste d'URL de flux RSS
foreach($res as $flux)
        {
                $arrayFlux[] = new fluxRss($flux["url"]);
        }
```

(2) : Pour chaque flux RSS créé, on fait ensuite appel à la fonction *chargementRSS* afin de récupérer les différents articles (sous format XML) publiés sur le flux qui seront stockés dans l'attribut *arrayArticles* sous forme d'objet articlesRSS. On va ensuite parcourir chaque article pour récupérer le titre, le lien et la description depuis leurs attributs.

```
foreach($array as $valeur)
	{
		$valeur->chargementRSS();

		if ($valeur->arrayArticles[0] != NULL) //test afin de vérifier que le flux RSS n'était pas
vide
		{
			foreach($valeur->arrayArticles as $valeur2)
			{
				$j++;

				$desc=$valeur2->getDescription();
				$lien=$valeur2->getLien();
				$titre=$valeur2->getTitre();
			...
			... 	// étape de préfiltrage
```

L'étape de pré-filtrage a lieu à ce moment-là. On se sert alors de notre dictionnaire de mots clés et des expressions régulières pour détecter si un de nos mots clés est présent dans le titre **et** la description. On récupère la liste de mots clés à l'aide d'une requête SQL basique :

```
SELECT * FROM Maladies&Symptomes
```

Pour effectuer nos requêtes vers la base de données, nous utilisons une classe MySQL qui a été écrite par Jocelyn DE GOER, un de nos tuteurs durant ce stage.

Cette classe dispose notamment de fonctions pour envoyer des requêtes pré-faites, comme par exemple dans notre cas, une requête *SELECT* basique.

```
$bd = new MySQL() ;

$keywords = $bd→requete_select_simple("Maladies&Symptomes"); //Il suffit seulement de préciser
le nom de la table pour réaliser une requête de type SELECT * FROM nomTable
```

La variable *$keyword* contient alors le résultat de la requête, soit un tableau avec un indice pour chaque colonne de la table.

On se sert alors des expressions régulières afin de déceler si le titre et la description contiennent un de nos mots clés.

La fonction *preg_match* de PHP permet de, en lui fournissant une expression régulière et une chaine de caractères à analyser, retourner la valeur 1 si jamais l'expression régulière correspond et de retourner 0 dans le cas inverse. En cas d'erreur, la fonction retourne la valeur FALSE.

Dans notre cas, nous allons parcourir la liste des mots clés et effectuer le test pour chaque article. Si l'article correspond, le programme passe à l'étape de nettoyage.

Si l'article ne correspond pas, il passe à l'article suivant.

```
…
… //suite de la boucle précédente
foreach($keywords as $keyword)
        {
        if( preg_match("#" .$keyword["keyword"]. "#",$titre) &&
preg_match("#" .$keyword["keyword"]. "#",$desc) ;
        //On rappelle que $titre contient le titre de l'article, $desc la
description et $keywords contient le résultat de la requête (avec la liste de mots clés)
        …
        … // étapes de nettoyage
```

(3) : On fait appel ici à la classe pageWeb et à sa fonction de nettoyage. On invoque le constructeur en lui donnant le lien de l'article afin qu'il récupère et stocke le code source de la page web contenant l'article. On invoque ensuite la méthode nettoyerPage() et on stocke le résultat, qui est donc le texte de l'article, dans une variable $str.

```
…
…
$page = new pageWeb($lien);
$str = $page->nettoyerPage();
…
… //insertion base de données
```

(4) : Il ne reste plus qu'à insérer le texte dans la base de données ainsi que le titre de l'article, son URL, la catégorie du mot clé trouvé et l'id du flux RSS duquel il a été extrait.

L'id de l'article dans la base de données sera les 10 premiers caractères du hachage MD5 du titre afin de faciliter son extraction plus tard et l'identifier par une chaine de caractères uniques. Le hachage en PHP s'effectue grâce à la fonction *md5()* .

On se sert de la classe MySQL pour insérer l'ensemble des valeurs dans la base de données grâce à la fonction *requete_insert* à laquelle on fournit un tableau contenant en indice les différentes colonnes à remplir.

```
…
…
```

```
                    $titre_md5_10 = substr(md5($titre),0,10); // la fonction substr() sert à extraire un segment de la chaine en paramètre

                $tab_valeurs = array("id" => $titre_md5_10,
                        "titre" => mysql_real_escape_string($titre),
                        "url" => mysql_real_escape_string($lien),
                        "texte" => mysql_real_escape_string($str),
                        "categorie" => mysql_real_escape_string($keyword["categorie"])
                    );

                $bd->requete_insert("articleWeb",$tab_valeurs);

                break ; //On termine la boucle par un break afin que le programme ne parcourt pas tout les mots clés alors qu'il en a déjà trouvé un.
            }
        }
```

(6) : Cette partie correspond à l'outil de text mining placé en aval. C'est lui qui va extraire les articles que notre script a aspiré et de déterminer si ils sont pertinents ou non. Cette partie ne fait pas l'objet de notre rapport.

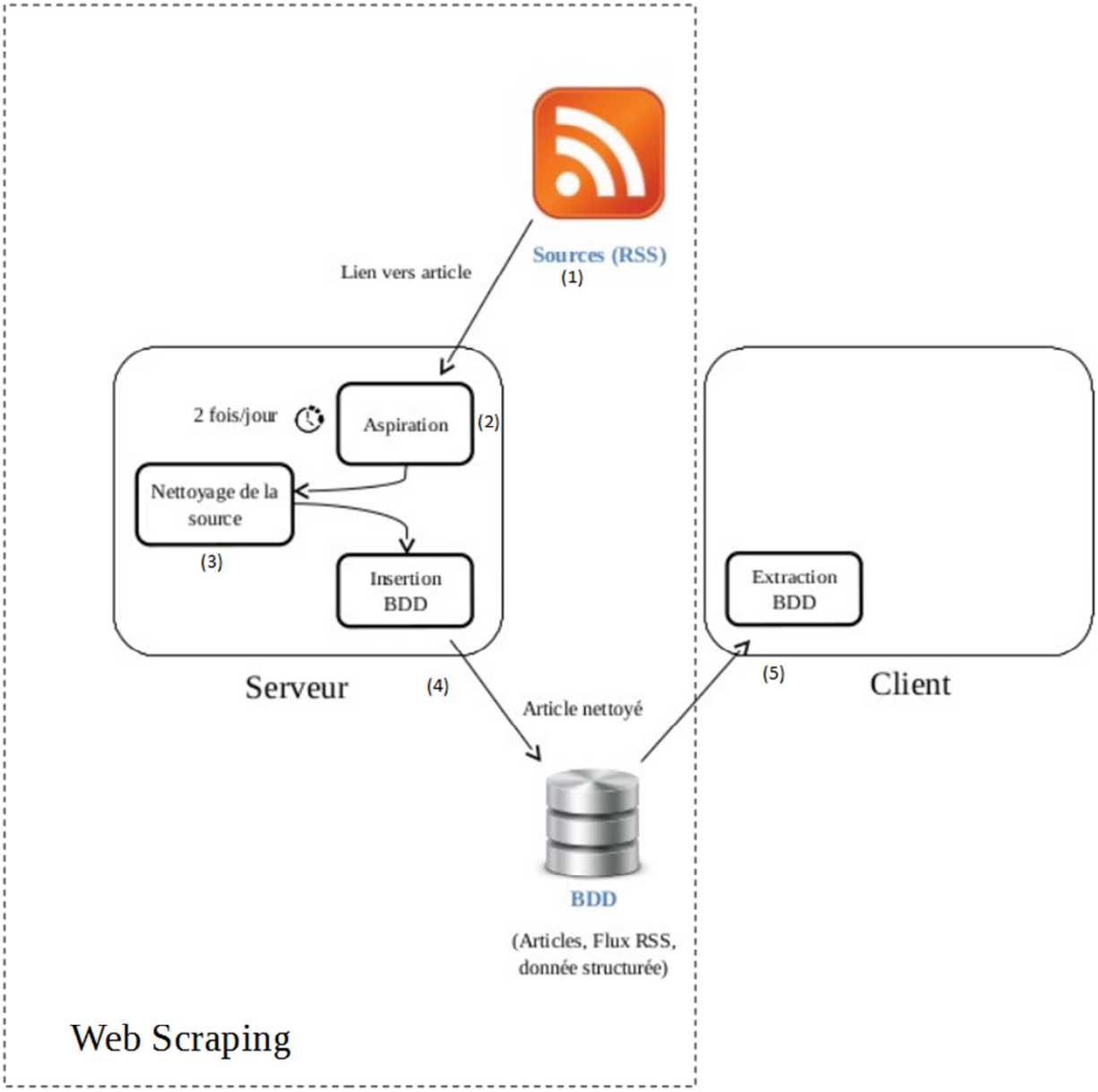


fig 3: Schéma architectural du déroulement général de la procédure

Rapport d'activité

Méthodologie

Au cours de ce projet, nous avons travaillé en utilisant les méthodes agiles de développement, grandement répandues dans le monde des projets informatiques.

Les méthodes agiles se basent sur 4 grands points que nous avons voulu respecter tout au long de ce stage au sein de notre unité au CIRAD.

- Travail d'équipe : Nous avons beaucoup travaillé et communiqué ensemble afin de suivre l'avancement de chacun dans le projet. J'ai également beaucoup travaillé avec mon collègue Clément Hemeury afin de s'entendre sur des parties que nous avions en commun.
- Application : Nous avons mis en avant le fonctionnement du programme par rapport au reste. La documentation étant une aide précieuse, elle n'a cependant pas constitué un but primaire.
- Collaboration : Nous avons impliqué le client dans le projet afin d'obtenir des retours directs tout au long de l'avancement du projet.
- Acceptation du changement : Nous avons adopté une planification et une structure logicielle flexible afin de permettre au projet d'évoluer dans le sens des besoins du client. Ces changements sont souvent réalisés grâce à la collaboration avec le client.

La communication sur l'avancement du projet avec nos tuteurs s'est faite de manière hebdomadaire, notamment à travers des fiches de rapports à rendre à chaque fin de semaine (fig 4).

Chaque semaine faisait aussi l'objet d'une réunion d'équipe, dans lesquelles les sujets comme les limites et les problèmes rencontrés étaient encouragés. Dans le cas où certains membres de l'équipe ne pouvaient pas directement assister à la réunion, une nouvelle réunion était organisée sur Skype.

CMAEE		Rapport d'avancement Hebdo	
		Equipe Epidémiologie	
Rédacteur :	BELOT Baptiste	Du :	10/08/2015
Liste de diffusion :		Au :	14/08/2015
1. Ce qui a été fait		**2. Ce qu'il reste à faire**	
Début de l'évaluation du script de nettoyage : Combien d'articles sont correctement nettoyé sur le nombre total d'articles récupéré		Evaluation à finir	
Filtrage par symptomes : si la maladie n'est pas trouvé dans le texte ou dans la description de l'article, alors on cherche les symptomes		Filtrage par symtomes à finir	
Rapport de stage : rédaction du rapport technique sur les technologies choisies et les flux RSS		Rapport de stage	
3. Problèmes rencontrés		**4. Prévu la semaine prochaine**	
Depuis l'implantation du filtrage par symptomes, le script vérifie 3 fois le même article.		Evaluation du nombre d'articles récupérés par rapport au nombre total (enregistrés et non enregistrés	
		Rapport de stage : suite du rapport technique	
		Filtrage par symtomes à finir	
		Début script pour donnée structurée	

fig 4: Exemple de rapport hebdomadaire

Limites et perspectives d'évolutions

La détection rapide et efficace de danger potentiel en santé animale et humaine est cruciale pour accélérer les mesures de riposte par les autorités sanitaires. Afin de répondre à ces nouveaux besoins, l'objectif de ce stage était de développer un outil permettant un accès plus rapide et automatique à des informations potentiellement cruciales pour la santé, améliorant un système global de surveillance numérique sur les potentiels dangers d'une épidémie virale, autant chez les humains que chez les animaux.

Des systèmes déjà existants entièrement automatisés jouent un rôle important dans le domaine de la VSI. Cependant, ces outils présentent certaines limites importantes comme notamment l'absence de contrôle et d'analyse dans leur processus de sélection, pouvant mener à des résultats incorrects et/ou non pertinents voir redondants.

Certains systèmes de la VSI sont quant à eux entièrement contrôlés par des experts et offrent des résultats extrêmement fiables. Seulement, ces systèmes peuvent souffrir de négligence et la sélection des résultats peut dépendre de la personne responsable de l'outil. D'autant plus que la modération d'un tel système prend énormément de temps comparé à un système automatique ce qui entraine des délais de réaction face aux épidémies émergentes mais aussi des coups supplémentaires (personnel à former et à embaucher).

Dans le cadre de ce stage, nous avons voulu créer un outil combinant les forces des outils

automatisés et celles des outils manuels afin d'obtenir un meilleur résultat à la fois sur la qualité des informations extraites et sur le temps de réaction. Notre système reste à la fois automatique mais est aussi soumis à un contrôle grâce à la liste de flux RSS et de mots clés sélectionnés par des humains.

Dans le cas de notre outil d'extraction (Web Scraping), nous pouvons y voir plusieurs optimisations afin d'améliorer d'autant plus les résultats.

- L'outil aurait aussi pu se consacrer à l'extraction de donnée sur les réseaux sociaux comme par exemple Twitter. Ces derniers étant en grande expansion et très réactifs sur les actualités, la publication d'informations est par conséquent extrêmement rapide. Dû à leur grande accessibilité, un filtre plus rigoureux aurait été nécessaire afin d'éliminer.
- Certains sites ont pour politique d'empêcher l'extraction de leurs informations. Ils ont par conséquent établi des systèmes de blocage de Web Crawler et autre outil automatique qui vise à extraire de l'information de page web. Il faudrait par conséquent investir du temps dans des méthodes de contournement de ces systèmes
- La fonction du nettoyage ne permet pas de garantir une extraction du texte de l'article à 100 %. En moyenne, nous avons évalué qu’environ 95 % des textes étaient correctement récupérés (test réalisé sur 254 articles, 241 étant bien extraits).
- Nous nous sommes rendu compte que le filtrage par symptômes a rendu le taux sélection plus haut par rapport à un filtre qui ne prend en compte que les maladies. Les conséquences ont été que certains articles extraits n’avaient presque rien à voir avec la santé animale ou les maladies émergentes. Nous aurions pu utiliser une méthode de combinaison de mots clés pour affiner les recherches et la sélection.

Notre système a l'avantage d'être semi-automatique et de pouvoir le diriger vers un sujet précis afin qu'il récupère des informations uniquement sur ce qui intéresse l'utilisateur. Dans notre cas, les chercheurs peuvent concentrer le logiciel sur les maladies dont ils souhaitent surveiller l'évolution dans le monde et avoir un rassemblement d'informations concernant ces maladies, ce qui constitue un énorme gain de temps.

Conclusion

Durant ce stage, nous avons donc pu participer à la Veille Sanitaire Internationale (VSI) et développer un outil d'acquisition de données textuelles lié à l'épidémiologie afin de répondre au besoin posé par la détection de maladies infectieuses et d'épidémies émergentes.

L'évaluation des informations extraites a fait preuve d'une grande précision et la récupération de ces informations s'est montrée très efficace. Les résultats ont donc l'air prometteur pour l'évolution et le futur de ce système.

Entre autres, le stage nous a permis de travailler en collaboration avec des épidémiologistes et en relation avec des domaines que nous ne connaissons pas. Cela nous a forcé à devoir expliquer des concepts de notre domaine à des personnes qui n'y sont pas familier.

Il nous a également permis de mettre en œuvre et d'étendre les connaissances acquises durant cette année spéciale à l'IUT de Montpellier. Le résultat de cette expérience est positive et récompensant, autant d'un point de vue professionnel, que d'un point de vue social et technique.

Résumé

Dans un contexte épidémiologique, nous devons créer un système permettant à des chercheurs participant à la Veille Sanitaire Internationale (VSI) de gagner du temps. Nous avons développé un outil d'extraction articles sur le web à partir de flux RSS. Nous utiliserons PHP pour pratiquer une discipline appelé le Web Scraping.

Mots clés : Web Scraping, flux RSS, PHP, VSI, maladies infectieuses émergentes