



UNIVERSITÉ DE MONTPELLIER

RAPPORT DE PROJET REDDIT SCRAPING

HLIN601

**Récolte de données
& Visualisation**

Auteurs:

Félix YRIARTE
Yanis HAMITOUCHÉ
Milissa DADI
David CAMARAZO

Encadrant:
Pascal PONCELET

*Nous tenons à remercier notre encadrant M. Pascal PONCELET
pour sa sollicitude et ses conseils avisés.*

Sommaire

1	Introduction	1
1.1	Objectifs du projet	1
1.2	Organisation du rapport	2
2	Outils utilisés	3
2.1	PRAW	3
2.2	Geonames	3
2.3	NLTK	3
2.4	MongoDB	3
2.5	Node-RED	4
2.6	OpenStreetMap et OpenLayers	4
3	Travail réalisé	5
3.1	Récupération de posts Reddit	5
3.2	Prétraitement	6
3.3	Heuristique	7
3.4	Tokenisation & POS-tagger	7
3.5	Geonames	7
3.6	Application Web	8
3.6.1	Page de vérification	8
3.6.2	Affichage de la carte finale	12
4	Organisation du travail	13
5	Bilan et conclusion	14
6	Bibliographie	15

1 Introduction

La récolte de données sur internet (*web scraping*) consiste à récupérer du contenu sur des sites web et à le réutiliser à d'autres fins. Cette technique d'extraction de données est régulièrement utilisé sur internet, par exemple :

- Les sites de comparaison de prix
- L'analyse de sentiments d'utilisateurs sur les réseaux sociaux

Une fois récupérées, il est possible d'y ajouter des informations.

C'est dans ce cadre que se situe le projet Reddit¹ Scraping : proposer une approche pour automatiquement récupérer des informations du site www.reddit.com afin de les analyser et de les enrichir.

Nous étions intéressés à l'idée de travailler sur ce sujet car la gestion et l'utilisation des données est une composante quotidienne dans l'informatique et l'utilisation qu'on peut en faire est très intéressante.

1.1 Objectifs du projet

Reddit est un regroupement d'une multitude de sous-forums. Dans ce sujet nous nous intéresserons à la partie www.reddit.com/r/EarthPorn où les utilisateurs publient une photo qu'ils ont avec un titre comme l'illustre la figure 1 :

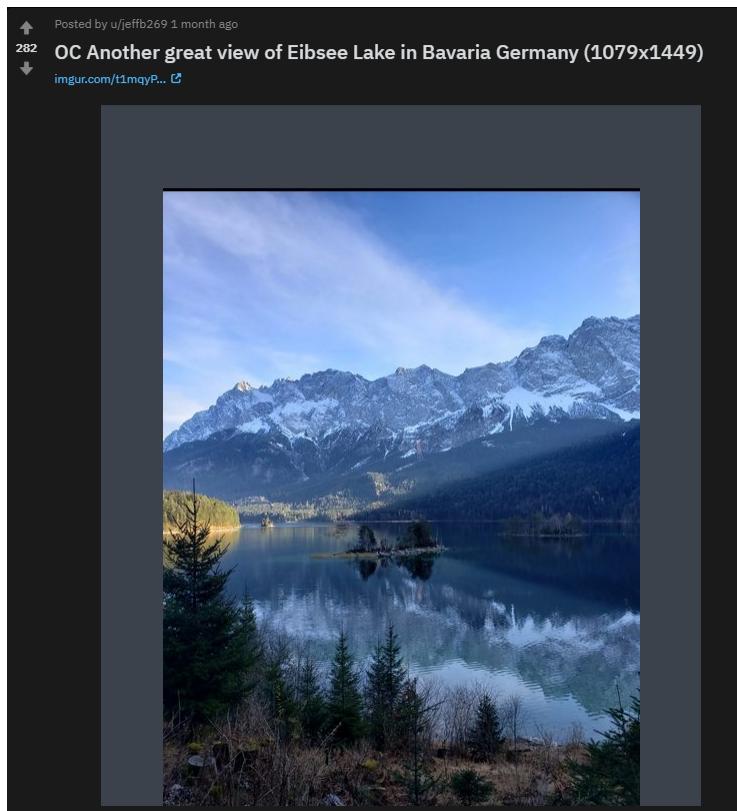


Figure 1: Une publication d'un utilisateur

Le but de ce projet est de récupérer toutes ces publications et de les afficher sur une carte du monde où nous pouvons cliquer et voir les informations. A première vue on peut voir qu'il va falloir récupérer le texte et l'analyser

¹Reddit est un site web communautaire d'actualités sociales fonctionnant via le partage de signets permettant aux utilisateurs de soumettre leurs liens et de voter pour les liens proposés par les autres utilisateurs.

afin d'obtenir des indicateurs de localisation. En naviguant sur Reddit on peut s'apercevoir que la tâche sera plus simple pour certaines publications que pour d'autres. Le titre du post présenté en figure 1 est concis et clair.

Sur la figure 2 illustrée ici, par exemple, le titre est excessivement long, et mentionne plusieurs endroits :

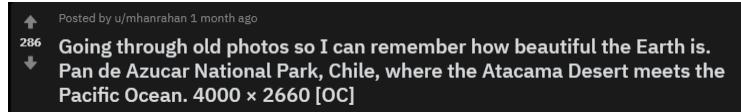


Figure 2: Un titre long d'un utilisateur

Bien sûr nous pourrions simplement lire le titre et placer à la main les photos sur une carte, nonobstant de ce côté rébarbatif, cela serait une solution. Cependant nous allons voir comment automatiser un peu plus ce processus grâce à plusieurs outils. Le code du projet est disponible sur le dépôt [github](#). Une vidéo de démonstration est visionnable sur [youtube](#).

1.2 Organisation du rapport

Le rapport est organisé de la manière suivante. Dans la section 2 nous présentons les outils qui ont été utilisés pour développer notre application. La section 3.1 décrit le processus pour connaître les coordonnées d'un lieu. Nous décrivons dans cette partie comment les posts Reddit sont récupérés, stockés et traités. Puis dans la section 3.6 concerne la plateforme de test et de visualisation. Nous présentons comment mettre en place des tests et afficher le résultat final sur une carte. Enfin nous concluons dans la section 5 le rapport en précisant les limites du travail ainsi que des remarques plus personnelles sur l'apport du projet.

2 Outils utilisés

2.1 PRAW

PRAW (*Python Reddit API Wrapper*) est une librairie python, c'est un *wrapper* (un emballage) de l'API Reddit. Il permet un accès facile à celle-ci. Une API (*Application Programming Interface*) est une interface standardisée et documentée qui permet à une application d'intégrer avec une autre.

2.2 Geonames

[Geonames](#) est une base de données géographique, on lui envoie une requête et il répond une liste de résultats possibles. C'est accessible via un site web classique mais aussi via API.

Montpellier		all countries	492 records found for "Montpellier"		
Name	Country	Feature class	Latitude	Longitude	
1 ⓘ Montpellier	France , Occitanie Hérault > Montpellier > Montpellier	seat of a second-order administrative division population 248,252	N 43° 36' 39"	E 3° 52' 38"	
2 ⓘ Montpellier Airport	France , Occitanie Hérault > Montpellier > Mauguio	airport elevation 5m	N 43° 34' 39"	E 3° 57' 30"	
3 ⓘ Mauguio	France , Occitanie Hérault > Montpellier > Mauguio	populated place population 18,320	N 43° 37' 5"	E 4° 0' 26"	
4 ⓘ Sète	France , Occitanie Hérault > Montpellier > Sète	populated place population 40,736, elevation 5m	N 43° 24' 10"	E 3° 41' 34"	
5 ⓘ Lattes	France , Occitanie Hérault > Montpellier > Lattes	populated place population 17,390	N 43° 34' 3"	E 3° 54' 16"	
6 ⓘ Frontignan	France , Occitanie Hérault > Montpellier > Frontignan	populated place population 22,231	N 43° 26' 54"	E 3° 45' 14"	

Figure 3: Une requête simple via le site web Geonames

2.3 NLTK

Natural Language Toolkit est une librairie python permettant l'analyse de texte. Cette librairie est souvent utilisée pour la classification, la tokenisation, l'étiquetage, etc.

2.4 MongoDB

Notre choix de base de données s'est très vite porté sur MongoDB. MongoDB est un SGBD² no-SQL très populaire dont les données sont représentées en JSON (ou BSON), nous permettant une manipulation facile des documents en Python et des API.

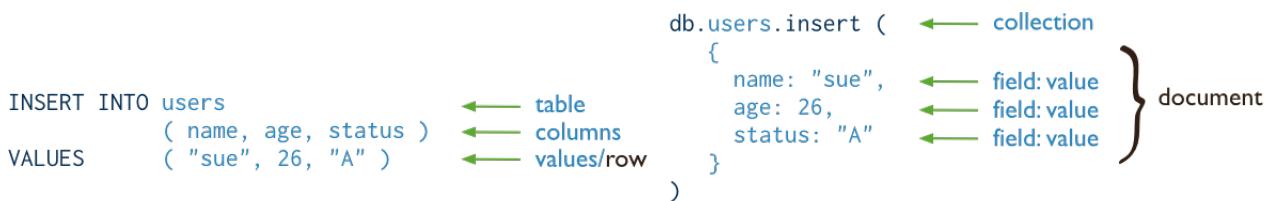


Figure 4: Un document inséré dans une base de données SQL, puis une base de données MongoDB

Pour notre projet, MongoDB était plus adéquat grâce à sa flexibilité et sa facilité d'utilisation. C'était aussi l'opportunité pour nous de découvrir ce SGBD noSQL juste après avoir suivi l'UE **Base de Données** le semestre précédent.

²Système de Gestion de Base de Données

2.5 Node-RED

Pour assurer la compatibilité entre chaque élément du projet, nous avons utilisé Node-RED. Cet outil de programmation permet de lier des noeuds contenant des fonctions JavaScript, le tout au sein d'un flow. Un *Message* (un objet JavaScript) parcourt le flow de noeud en noeud, et se charge en informations au fur et à mesure. Node-RED nous permet d'effectuer des requêtes HTTP, de récupérer leur résultat, puis de l'envoyer dans des scripts python ... et est ainsi essentiel dans ce projet. C'est également avec Node-RED que nous avons mis en place la page de vérification des posts, et celle répertoriant les posts acceptés. Des captures d'écran des flows sont disponibles en annexe : figure 13. Plus d'informations quant à Node-RED sont disponibles sur <https://nodered.org/>.

2.6 OpenStreetMap et OpenLayers

Les cartes que nous utilisons sont importées de [OpenStreetMap](#). Nous utilisons également [OpenLayers](#) qui ajoute des couches sur les cartes utilisées, nous permettant ainsi d'y placer des pointeurs représentant les posts géolocalisés.

3 Travail réalisé

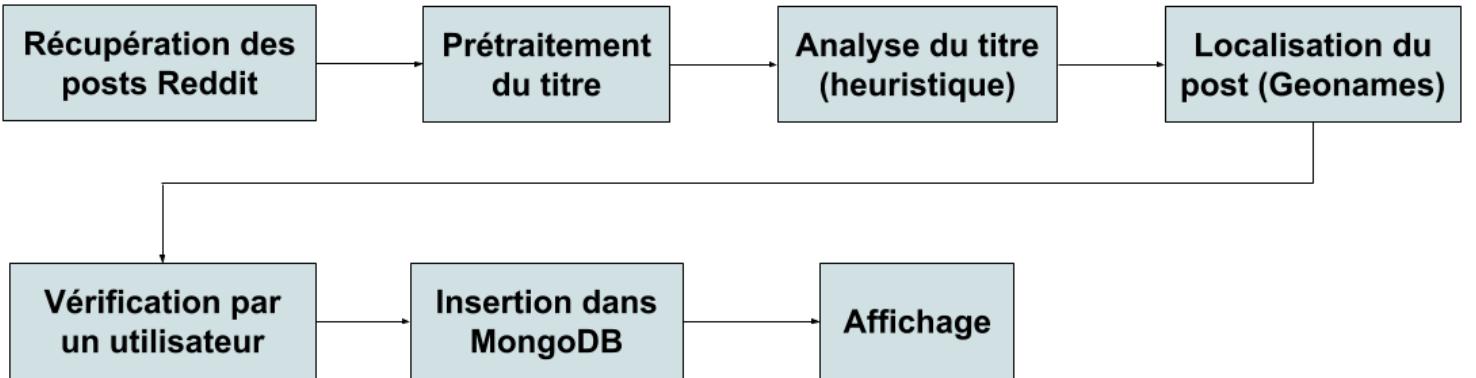


Figure 5: Vue d'ensemble du projet

Le fonctionnement général de notre application est décrit dans la figure 5. Les posts Reddit obtenus de la plateforme sont tout d'abord traités par des fonctions d'heuristiques. Une requête à Geonames est effectuée ; on récupère une localisation. Enfin nous utilisons l'application web pour vérifier les posts par les utilisateurs : Les posts sont alors insérés dans la MongoDB ; nous pouvons afficher les posts validés sur une carte.

Pour nous laisser la possibilité de modifier certains morceaux du projet à tout moment, nous avons décidé de travailler avec Node-RED. En effet, cela nous a permis de conserver une forte atomicité : chaque fonction de notre projet est un programme à part entière. Nous pouvons, grâce à Node-RED, lier chacune de ces fonctions. Les différentes fonctionnalités sont décrites en détail dans les sous sections suivantes.

3.1 Récupération de posts Reddit

Pour récupérer les posts sur le subreddit *EarthPorn* nous avons utilisé la librairie python **PRAW**. L'API Reddit permet beaucoup de choses, mais ici nous serons seulement intéressés par les posts d'un endroit précis.

Source Code 1: Exécuté côté serveur, ce code permet d'enregistrer dans une variable les informations des posts

```

ep_subreddit = reddit.subreddit('EarthPorn')

for post in ep_subreddit.hot(limit=30):
    posts = []
    posts.append([post.title, post.score, post.id, post.url])
    posts = pd.DataFrame(posts,columns=['title', 'score', 'id','url'])
    postsFormate = posts.to_dict(orient='list')

```

Nous pouvons récupérer plusieurs données d'un post, comme le titre, le score, l'id et l'url de la photo. D'autres peuvent être intéressants notamment pour proposer des filtres comme la date de création du post, le nombre de commentaire, etc. La figure 6 illustre le fichier JSON envoyé par l'API pour un post Reddit.

```
{"title": ["Sunrise at over the Buffalo National River. (OC) (1086x724)"], "score": [5163], "id": ["geaq32"], "url": ["https://i.redd.it/hl7bpocrslx41.jpg"]}
```

Figure 6: Le résultat de l'appel à l'API Reddit

3.2 Prétraitement

Une fois les posts enregistrés dans la base de données, on peut s'intéresser au titre. Pour avoir des coordonnées d'un lieu il faut questionner Geonames. Donc nous allons envoyer le titre du post à Geonames, mais comme nous avons pu le voir dans la figure 1, il n'y a pas que du texte dans ces titres, il y a des indésirables, comme la résolution de la photo et autre tag [OC]. Si on envoie des caractères non communs, des chiffres, de la ponctuation, Geonames ne saura pas répondre. Il faut donc effectuer un prétraitement.

Le prétraitement va servir à retirer ce qui est forcément inutile, donc pas du texte pur qui peut contenir des éléments de localisation. En survolant un peu les posts des utilisateurs on se rend compte qu'il y a plus ou moins une façon admise sur comment écrire les titres, avec notamment la résolution de la photo (par ex: [1080x1350]). Il y a aussi la plupart du temps un tag OC³ et parfois un compte de réseau social. Avec une fonction de prétraitement, si on a un titre original comme :

Bavarian Alps [OC] [1280x1920] IG @holysht0t

On peut passer à :

Bavarian Alps

Pour certains posts cela peut suffire, si l'utilisateur écrit seulement le lieu de la photo, Geonames donnera directement le bon résultat, comme dans la figure 7 avec Bavarian Alps :

Bavarian Alps		all countries	
		search [advanced search]	
1 records found for "Bavarian Alps"			
Name		Country	Feature class
1 🇦ustria	Bavarian Alps 🌎	Austria	mountains
	Alpes bavaroises, Alpi bavaresi, Bavarian Alps, Bayerische Alpen, Bayrische Alpen, North Tyrol Alps, Tyrol...		N 47° 30' 0" E 11° 0' 0"

Figure 7: Le résultat de la recherche avec titre prétraité

³Original Content : la photo a été prise par le créateur du post Reddit

3.3 Heuristique

Une heuristique est une méthode pour déterminer un résultat. Si nous envoyons trop de mots ou pas assez ou encore pas les bon à Geonames, il ne nous donnera pas de résultat correct. En effet il peut y avoir plusieurs lieux avec le même nom, ou des ressemblances. Geonames a besoin des meilleurs candidats possible pour renvoyer le résultat le plus précis.

Un titre trop long comme nous avons pu voir en figure 2 ne nous renverra pas de résultat.

Going through old photos so I can remember how beautiful the Earth is.

Pan de Azucar National Park, Chile, where the Atacama Desert meets the Pacific Ocean.

Il nous faut donc une fonction qui donne les meilleurs candidats à envoyer à Geonames, pour se faire on va réaliser une analyse morphosyntaxique.⁴

3.4 Tokenisation & POS-tagger

La tokenisation (tokenization en anglais) est le fait de segmenter un texte en unités atomiques: les tokens. Il s'agit le plus souvent du découpage en mots ou en phrases.

Exemple: la phrase "**L'homme sage pardonne sans bruit.**"

deviendrait : "L", "", "homme", "sage", "pardonne", "sans", "bruit", ".".

Un POS-tagger (part-of-speech tagger) s'occupe de catégoriser les mots. En français, on parle d'étiquetage morphosyntaxique ou étiquetage grammatical. Ainsi, chacun des tokens précédemment obtenus sera associé à un tag, correspondant à la classe grammaticale du mot (nom, déterminant, adjetif, verbe, pronom...).

Source Code 2: Une phrase tokénisée ou étiquetée grâce aux fonctions fournies par la librairie NLTK

```
>>> text = word_tokenize("Another great view of Eibsee Lake in Bavaria Germany")
>>> nltk.pos_tag(text)
[('Another', 'DT'), ('great', 'JJ'), ('view', 'NN'), ('of', 'IN'), ('Eibsee', 'NNP'),
('Lake', 'NNP'), ('in', 'IN'), ('Bavaria', 'NNP'), ('Germany', 'NNP')]
```

Où *DT* = déterminant, *JJ* = adjetif, *NNP* = nom propre, etc.

Cette catégorisation nous permet d'éliminer facilement tous les mots qui ne nous intéressent pas.

Comme première heuristique nous avons choisi de garder seulement les noms propres *NNP* car les noms des lieux sont le plus souvent des noms propres avec une majuscule, le pos-tagger va reconnaître ces majuscules.

3.5 Geonames

Une fois que nous avons les bons candidats, les données de la base sont alors extraits afin de pouvoir déterminer leur géolocalisation grâce à Géonames. En python, grâce à l'API nous pouvons questionner Geonames et il nous renvoie un JSON des réponses :

Source Code 3: API Géonames

```
req = requests.get('http://api.geonames.org/searchJSON?username=rheiloth&maxRows=1&q={0}'
    .format(titreFormat))
```

⁴Analyse de texte qui consiste à évaluer la forme et la fonction de ses éléments constitutifs.

Avec *Eibsee Lake Bavaria Germany* l'API nous donne :

```
{"TotalResultsCount":3,"geonames": [{"adminName1": "Bavaria",  
"countryName": "Germany",  
"fcodeName": "lake",  
"name": "Eibsee",  
"lat": "47.45601",  
"lng": "10.97311",...},  
{  
"adminName1": "Bavaria",  
"countryName": "Germany",  
"fcodeName": "populated place",  
"name": "Eibsee",  
"lat": "47.45636",  
"lng": "10.99053",...},  
...]}]
```

Comme nous pouvons le voir, nous obtenons plusieurs résultats proches. Pour choisir lequel est le plus pertinent, on vérifie d'abord si l'un des mots présents dans la string candidate est également présent dans le "fcodeName" renvoyé. Sinon, nous prenons le premier élément de la liste de localisation.

Malgré ces précautions pour avoir le bon résultat il se peut qu'il y ait des erreurs ou une incapacité à trouver une localisation. C'est pourquoi nous avons créé une plateforme en ligne où l'utilisateur peut vérifier et aider à la localisation.

3.6 Application Web

L'application a été réalisé en utilisant Node-RED qui est un outil de développement visuel qui permet de relier des composants.

3.6.1 Page de vérification

Nous avons mis en place une page de vérification. Elle permet à l'utilisateur de voir l'image du post, son titre épuré (c'est à dire après passage dans la fonction de prétraitement), et, si une géolocalisation a été proposée, une carte centrée sur un pointeur placé à la localisation en question. Cette page offre en fait 2 usages : elle sert à la fois de vérification de la localisation (si il y en a une), et d'aide à l'amélioration de notre heuristique dans le cas contraire. En effet, l'utilisateur peut cliquer sur les mots du titre qui, d'après lui, servent à localiser l'endroit. Ces mots sont alors ajoutés à l'objet qui sera par la suite enregistré dans la mongoDB.

Par exemple, pour notre exemple précédent, on obtiendrait :

NEXT CLEAR switch

titre epure : Another great view of Eibsee Lake in Bavaria Germany



Figure 8: Page de vérification des posts avant entrée dans la mongoDB

L'utilisateur a passé le switch à True, celui ci devient donc bleu : d'après l'utilisateur, la géolocalisation correspond bien. L'objet contiendra alors un booléen passé à True lors de son insertion dans la MongoDB.

Ce n'est cependant pas toujours le cas. Quand l'utilisateur pense que la géolocalisation ne correspond pas au titre (ou à l'image), il peut le faire savoir :

NEXT CLEAR switch

titre épure : Sunset over the **Eastern Sierras** from an alpine meadow at' elevation

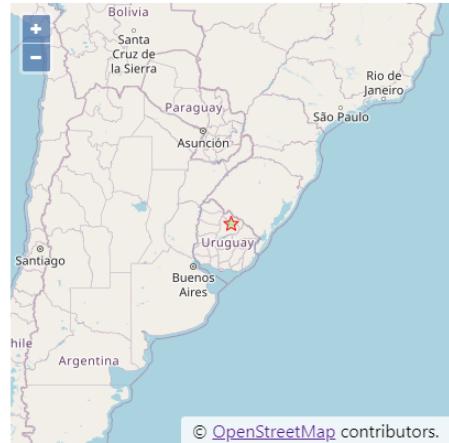


Figure 9: Une mauvaise prédiction

Ici, l'utilisateur a cliqué sur les mots *Eastern Sierras* (qui s'affichent en rouge). Ces mots sont alors ajoutés à l'objet, en plus du booléen False (car la géolocalisation ne correspond pas au titre. L'utilisateur n'a donc pas basculé le switch). Ce post ne sera finalement pas affiché dans le répertoire final, mais sera stocké dans la mongoDB malgré tout, nous permettant ainsi d'améliorer l'heuristique.

Enfin, même si aucune localisation n'a été trouvée, l'utilisateur peut signaler les mots qui lui semblent utiles pour trouver l'information :

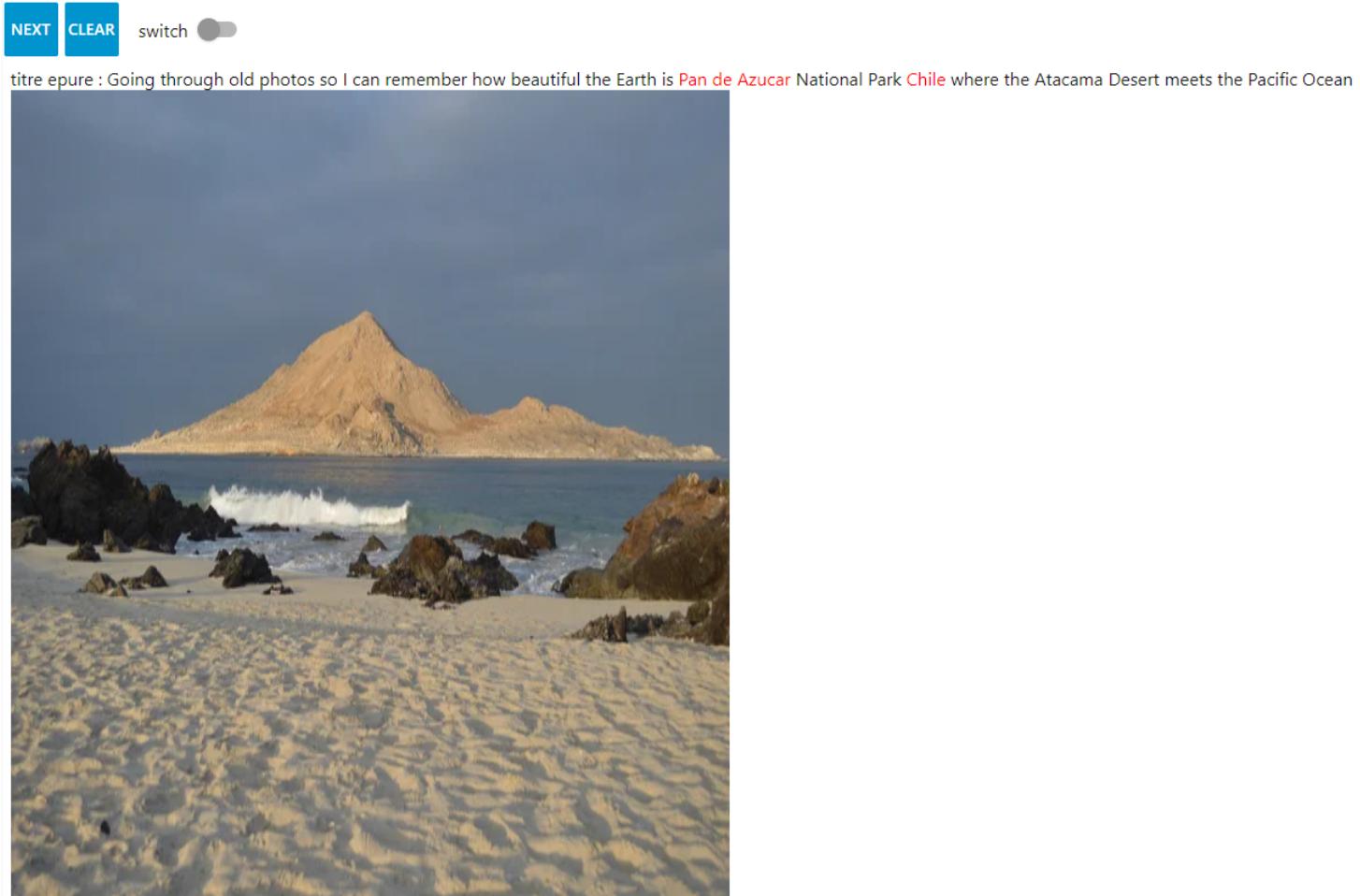


Figure 10: Un post pour lequel aucune prédiction n'a été trouvée

3.6.2 Affichage de la carte finale

Enfin, une page HTML a été créée pour visualiser les posts ayant été validés. Lors du chargement de la page, une *XMLHTTP request* récupère les objets validés dans la mongoDB. Avec la librairie OpenLayers, on peut alors les afficher sur la carte OpenStreetMap. Un click sur un marqueur affiche l'image du post ainsi que son titre, en plus d'un lien vers le post original. Cette page est disponible [ici](#).

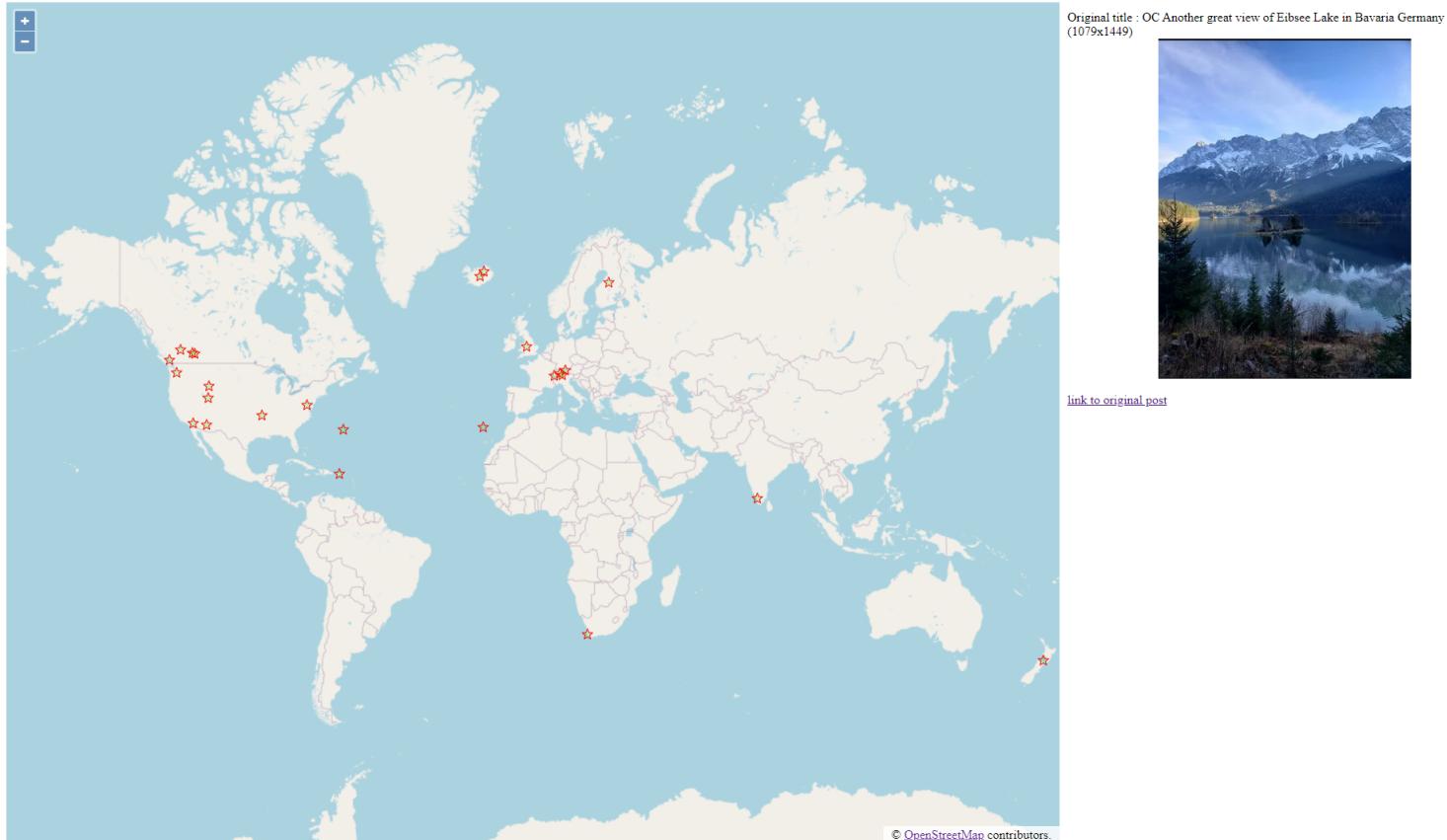


Figure 11: Répertoire des posts acceptés

4 Organisation du travail

Le projet **Reddit Scraping** a été réalisé par Félix Yriarte, Yanis Hamitouche, Milissa Dadi et David Camarazo dans le cadre de l'UE **HLIN601 - Projets de Programmation de L3**, de février à mai 2020.

Nous avons été supervisés par Pascal PONCELET, avec qui nous avons eu rendez-vous toutes les deux semaines pour lui montrer nos progrès et faire le point sur nos prochains objectifs. A la fin de chaque réunion nous faisions un compte rendu comme dans la figure 16.

Ce diagramme de Gantt donne un bon aperçu des tâches effectuées au cours du semestre :

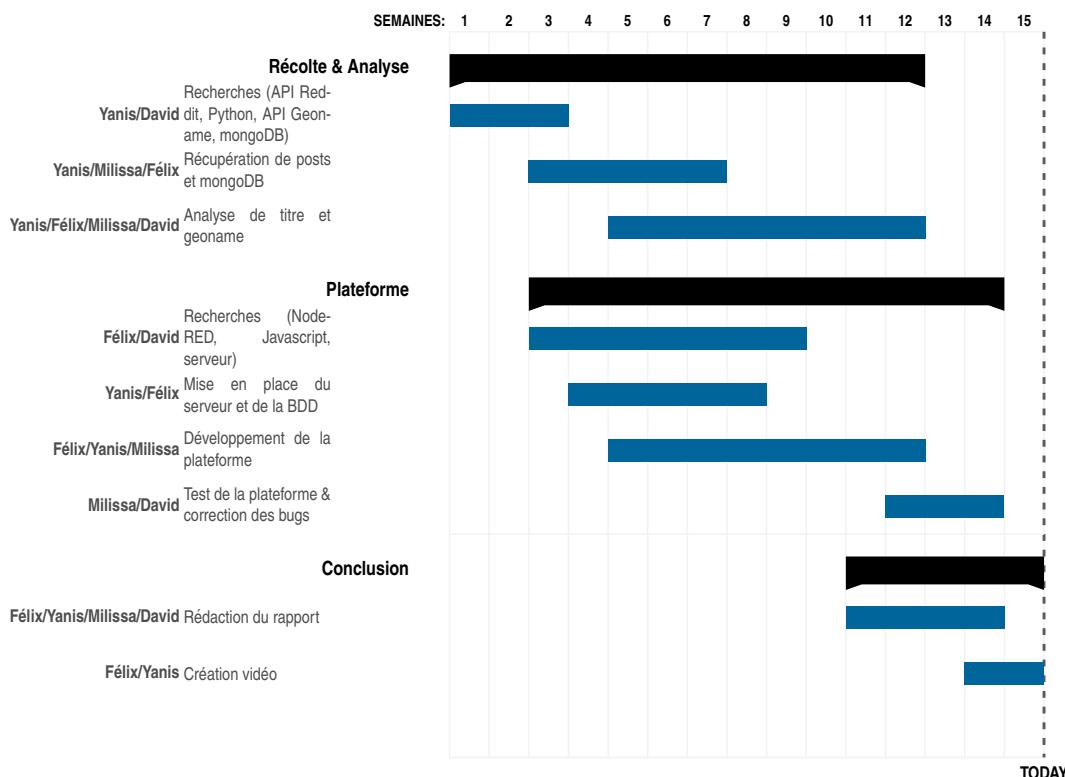


Figure 12: Gantt

5 Bilan et conclusion

Le projet s'est déroulé en deux parties, d'une part la récolte et l'analyse de posts Reddit, nous avons récupéré les posts, traité les titres, demandé à Geonames la localisation et stocké le résultat ; et dans un second temps nous avons pu afficher ce résultat sur une carte dans une application en ligne avec la possibilité de vérifier et aider à la localisation.

La carte répertoriant les posts acceptés est accessible [ici](#).

Une vidéo de démonstration de l'application est disponible sur [youtube](#).

Et enfin le lien [github](#) du projet.

Nous avons appris à faire face aux difficultés rencontrées dans un domaine jusqu'alors inconnu pour nous (**data mining**) mais avons eu la chance d'avoir pu bénéficier de l'aide et des conseils de notre encadrant tout au long de la réalisation du projet, lors de nos rendez-vous le jeudi ou par email quand c'était nécessaire.

Nous sommes très satisfaits d'avoir pu atteindre les objectifs du projet. Du temps supplémentaire nous aurait toutefois permis d'améliorer le système d'heuristique et d'ajouter des fonctionnalités à la page web.

6 Bibliographie

API Reddit : <https://www.reddit.com/dev/api/>

PRAW : <https://praw.readthedocs.io/en/latest/>

API Geonames : <http://www.geonames.org/export/web-services.html>

NLTK : <https://www.nltk.org/>

Node-RED : <https://nodered.org/>

MongoDB : <https://docs.mongodb.com/>

OpenStreetMap : <https://www.openstreetmap.org/help/>

Annexes

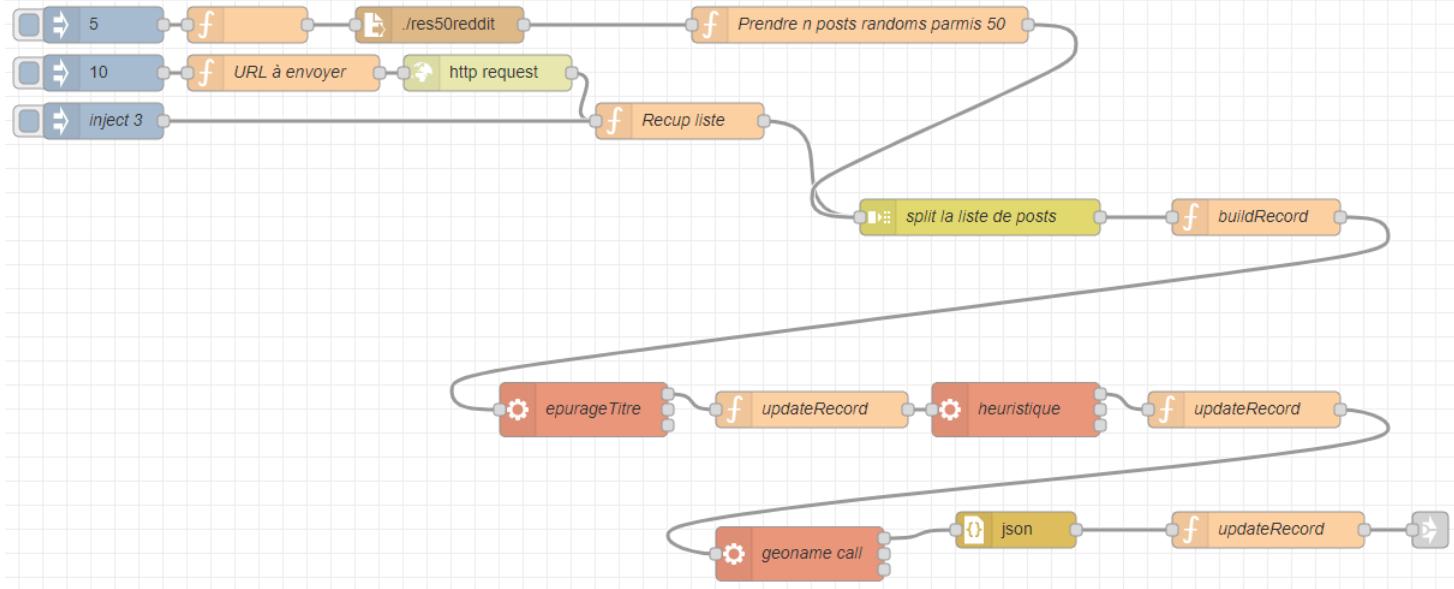


Figure 13: Flow Node-RED effectuant les appels aux APIs et le traitement du titre

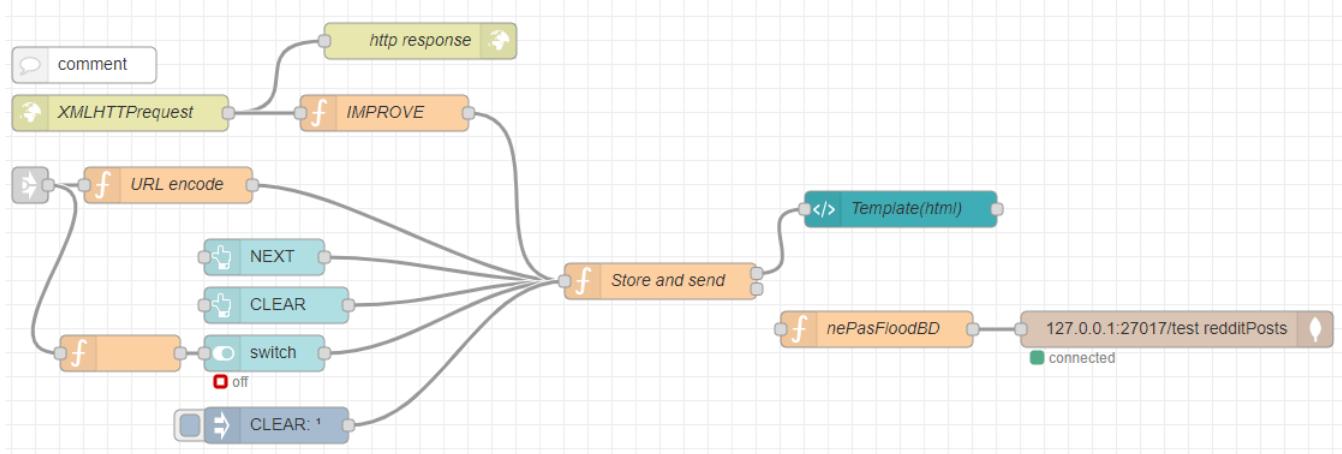


Figure 14: Flow Node-RED de la page vérification

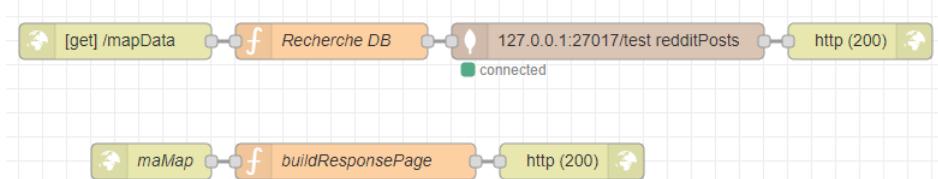


Figure 15: Flow Node-RED correspondant à la page map

CR Réunion du [jeudi 27 février 16h00](#)

Lieu : département informatique

Présent(s) : David Camarazo, Felix Yriarte, Milissa Dadi, Yanis Hamitouche

Points abordés:

- * heuristique par combinaison de candidats
- * affichage sur carte avec leaflet

Décision :

- * algorithme de décision du meilleur candidat par combinaison, retourne la combinaison qui a le moins de résultat
- * (?) petite démo résultat sur carte

Documents/Codes échangés

- * Aucun

Date et heure prochaine réunion : [jeudi 12 mars 14h30](#)

Lieu prochaine réunion : département informatique

Figure 16: Compte rendu réunion