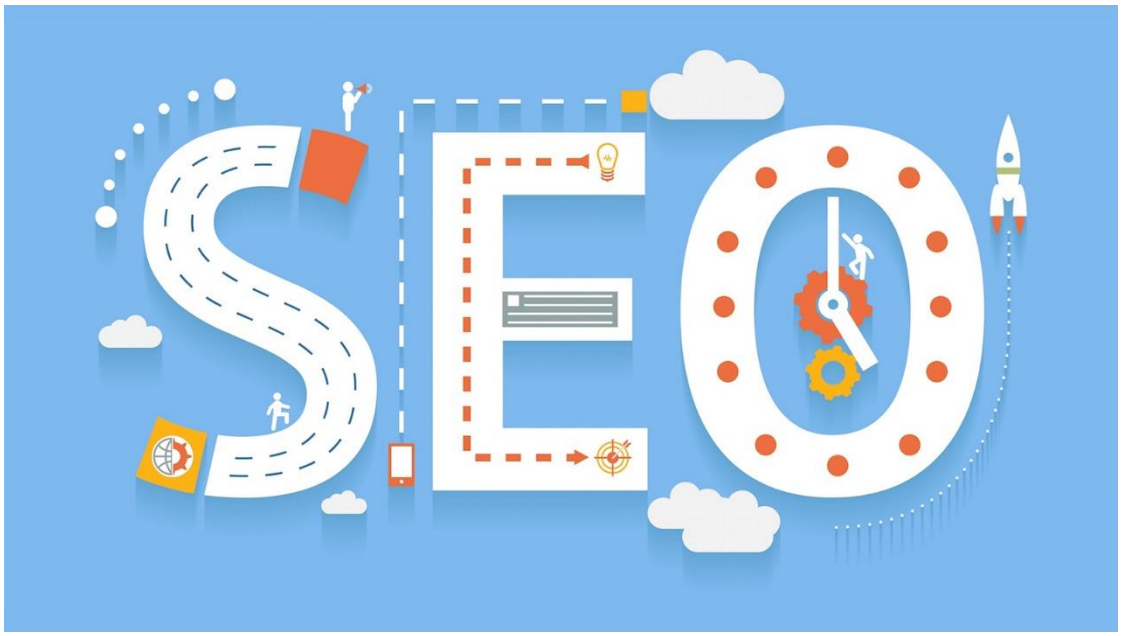
Université de Tours

Polytech Tours

MISE EN OEUVRE D'OUTILS SEO POUR LA PREDITCION DE TRAFIC WEB

PROJET DÉVELOPPEMENT



SYMIEC Marceau

LACOSTE Pierre

DII3 – Promotion 2021/2024

M. DELALANDRE Mathieu

Remerciements

Nous souhaitons remercier M. DELALANDRE, notre tuteur, pour avoir su libérer du temps afin de nous accompagner et nous aider durant ce projet. Il a en effet toujours été présent pour répondre à nos questions et nous prodiguer de nombreux conseils.

SOMMAIRE

1 Introduction

Le SEO définit le processus consistant à rendre un site Web plus visible sur la page de résultats d'un moteur de recherche. Il est donc important pour une personne possédant un site Web de définir une stratégie de référencement afin de placer son site web en haut de la liste d'une page de recherche. Cela augmente ainsi la probabilité que les gens visitent le site Web.

Une partie importante du SEO consiste à comprendre l'importance des mots-clés. Les mots clés sont des mots ou des ensembles de mots spécifiques qui décrivent le mieux le thème ou le concept général d'une idée, d'une page produit, d'une entreprise, ou d'un site Web. Il est donc important de choisir les bons mots clés pour améliorer son classement SEO et ainsi générer du trafic vers son site web.

Pour l'amélioration de son classement SEO, d'autres éléments sont aussi à prendre en compte. En effet, les liens, les titres, les méta descriptions, le temps de chargement de la page web, l'originalité du contenu, etc. sont aussi des points importants.

Pour notre projet nous nous concentrerons sur les mots clés dans le but d'améliorer le référencement SEO. Pour cela, nous allons extraire les mots clés de plusieurs sites web traitant d'une même thématique et étant bien référencé par un moteur de recherche pour connaître les mots clés sur lesquelles s'appuyer afin de générer du trafic. Pour finir, nous utiliserons google trends pour analyser ces mots clés et n'utiliser que les plus pertinents, c'est-à-dire, ceux possédant le plus gros volume de requête.

2 Extraction des mots clés

2.1 Le modèle BERT

Pour l'extraction, nous nous sommes dirigés vers un modèle de deep learning pré-entraîne, appelé BERT, qui actuellement, semble être la solution offrant les meilleurs résultats pour une grande partie des tâches NLP. Ce modèle a été publié par la filière d'intelligence artificielle de Google en 2018.

Le modèle BERT est pré-entrainé de manière non supervisée, c'est-à-dire qu'il ne nécessite pas de jeu de données labellisé. Ce pré-entrainement est fait en suivant 2 techniques. La première est appelée « Masked ML (MLM) ». Cette méthode consiste à masquer aléatoirement des mots dans une phrase, puis tenter de les prédire. Pour prédire ces mots, le modèle regarde dans les deux sens de la phrase en utilisant le contexte complet de celle-ci, afin de prédire le mot masqué.

Pour illustrer cela, prenons l'exemple de la séquence initiale suivante : « La personne va au supermarché et achète une paire de chaussures ». La séquence donnée sera « La personne va au supermarché et achète une _ _ _ paire de chaussures ». Ici, le modèle devra donc prédire le mot « paire ».

La deuxième méthode est appelée, « Next Sentence Prediction (NSP) ». Elle consiste à prédire si, une certaine séquence A est suivi par une certaine séquence B. Prenons l'exemple suivant avec une séquence A : « Il va pleuvoir. » et une séquence B « Je prends mon parapluie ». Ici, le modèle doit prédire si la séquence A est bien suivie par la séquence B et mettre le label « IsNext » si c'est vrai et dans le cas contraire « NotNext ». Cela permet donc à BERT de mieux comprendre les interdépendances entre les phrases qui se suivent.

2.2 La bibliothèque KeyBERT

Pour l'extraction des mots clés, nous avons donc utilisé KeyBERT, une bibliothèque d'extraction de mots clés qui exploite les intégrations BERT pour obtenir les mots clés les plus représentatifs du document texte.

L'image ci-dessous représente le fonctionnement général de keyBERT à un niveau très élevé.

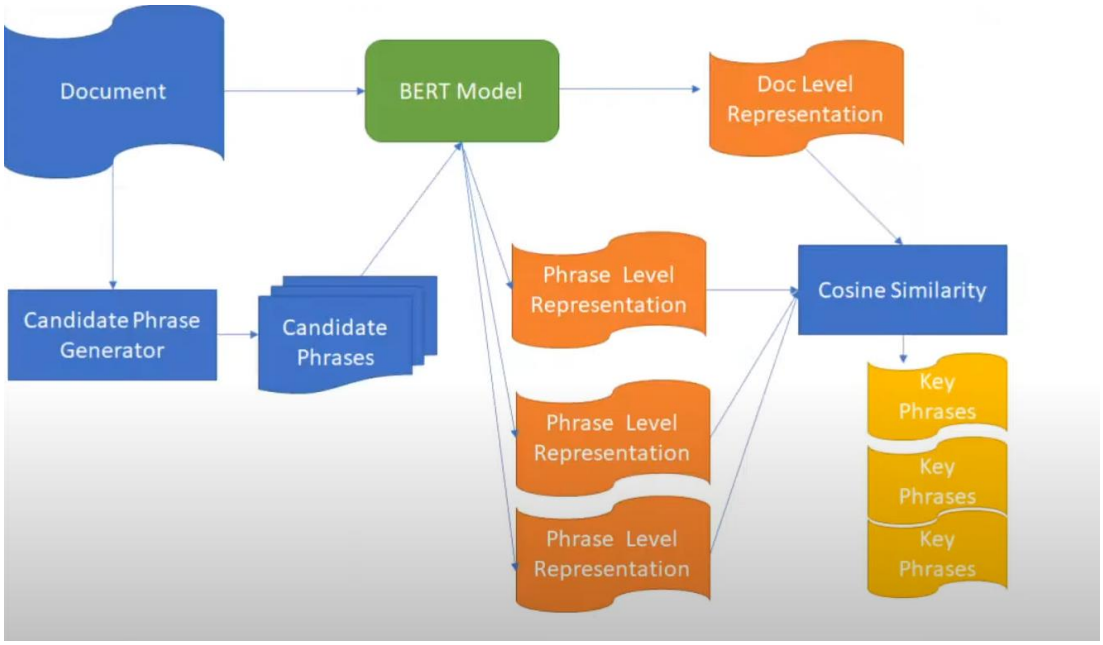


Figure 1 : Schéma du fonctionnement général de KeyBERT

La bibliothèque keyBERT fonctionne selon 3 étapes. La première étant l'extraction des « n-grammes » du texte. Les « n-grammes » ne sont rien d'autre que la séquence de n éléments consécutifs dans une chaîne de caractères. KeyBERT utilise CountVectorizer pour obtenir une liste de n-grammes candidats. La similarité de chaque phrase clé avec le document est ensuite mesurée à l'aide de la similarité cosinus. Et pour finir, les mots les plus similaires peuvent alors être identifiés comme les mots qui décrivent le mieux l'ensemble du document et sont considérés comme des mots-clés.

2.3 La similarité cosinus

La similarité cosinus est une mesure qui quantifie la similitude entre deux vecteurs ou plus. Cela représente le cosinus de l'angle entre les vecteurs.

La similitude cosinus est décrite mathématiquement comme la division entre le produit scalaire des vecteurs et le produit des normes euclidiennes ou de la grandeur de chaque vecteur.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 2 : Formule de la similarité cosinus

Prenons un exemple, pour mieux comprendre ce que cela représente. Sur le graphique ci-dessous l'angle entre les vecteurs A et B est de 10 degrés. Le cosinus de l'angle entre ces vecteurs est donc : Cos (10) = 0,98. Selon cette méthode, on pourrait dire que les angles sont similaires à 98%.

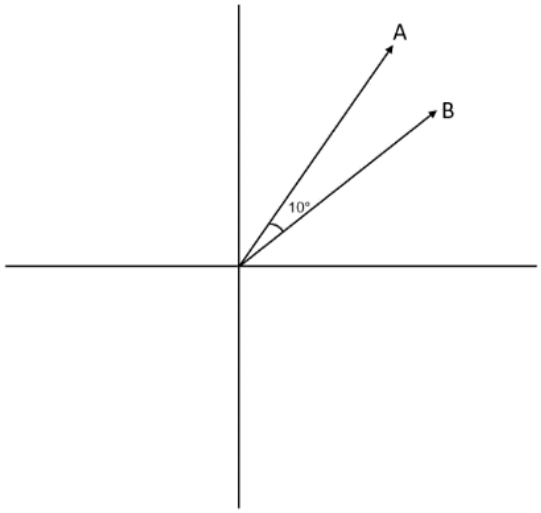


Figure 3 : Graphique illustrant la similarité cosinus

En appliquant cela à un document, les phrases de celui-ci sont converties en une forme de représentation vectorisée. Le calcul du cosinus est donc effectué entre les vecteurs représentant les phrases et indique le pourcentage de similitude entre les phrases des documents.

2.4 Mise en œuvre

Pour l'extraction des articles, nous avons utilisé la bibliothèque « newspaper3k » qui permet, à partir d'une URL, d'extraire différentes informations tels que le texte, les images, l'auteur ou la date de publication. Pour notre application nous avons simplement extrait le texte des articles afin de pouvoir en tirer des mots clés pertinents.

Pour l'extraction des mots clés, nous avons donc utilisé la bibliothèque « KeyBERT ». Nous avons créé une instance de KeyBERT acceptant un paramètre, qui est le modèle Sentences-Bert. Le choix du modèle pré-entraîné est important, car il va définir la pertinence des mots clés ainsi que performance de recherche. Nous avons donc choisi un modèle spécialement entrainé pour la recherche sémantique.

Model Name	Performance Sentence Embeddings (14 Datasets)	Performance Semantic Search (6 Datasets)	Avg. Performance	Speed	Model Size
all-mpnet-base-v2	69.57	57.02	63.30	2800	420 MB
multi-qa-mpnet-base-dot-v1	66.76	57.60	62.18	2800	420 MB
all-distilroberta-v1	68.73	50.94	59.84	4000	290 MB
all-MiniLM-L12-v2	68.70	50.82	59.76	7500	120 MB
multi-qa-distilbert-cos-v1	65.98	52.83	59.41	4000	250 MB
all-MiniLM-L6-v2	68.06	49.54	58.80	14200	80 MB
multi-qa-MiniLM-L6-cos-v1	64.33	51.83	58.08	14200	80 MB
paraphrase-multilingual-mpnet-base-v2	65.83	41.68	53.75	2500	970 MB
paraphrase-albert-small-v2	64.46	40.04	52.25	5000	43 MB
paraphrase-multilingual-MiniLM-L12-v2	64.25	39.19	51.72	7500	420 MB
paraphrase-MiniLM-L3-v2	62.29	39.19	50.74	19000	61 MB
distiluse-base-multilingual-cased-v1	61.30	29.87	45.59	4000	480 MB
distiluse-base-multilingual-cased-v2	60.18	27.35	43.77	4000	480 MB

Figure 4 : Liste de modèles pré-entraînés

À la suite de plusieurs essais avec des modèles différents, nous avons opté pour le modèle « all-mpnet-base-v2 » qui semble être le modèle le plus performant. Pourtant ce modèle n'a pas été entrainé sur des textes français et parait plus performant qu'un modèle multilingue ou français tel que CamemBERT ou FlauBERT.

Depuis la bibliothèque KeyBERT, nous utilisons la fonction « extract_keywords » qui accepte plusieurs paramètres tels que le texte, le nombre de mots qui composent la phrase clé et le nombre de mots clés à récupérer.

De plus, après plusieurs essais, nous avons décidé de ne pas utiliser la similitude de la somme maximale ainsi que la pertinence marginale maxime (MMR), car lorsqu'on opte pour la diversité, on perd beaucoup de pertinence dans les résultats. Nous avons donc utilisé la méthode native qui semble mieux fonctionner en extrayant des mots clés toujours très pertinents par rapport au sujet.

Pour finir, pour maximiser, la pertinence des résultats nous avons utilisé une liste de « stop words », qui sont des mots n'ayant pas de réelle signification. Ce sont des mots courants et qui reviennent de façon tellement régulière qu'ils ne permettent pas de caractériser, au sens lexical, un texte par rapport à un autre texte. Les « stop words » sont généralement des mots dits grammaticaux tels que des adverbes, des pronoms ou encore des mots de liaison.

3 Google Trends

Google trends est un outil proposé par google permettant de regarder sur la France entière ou par régions (anciennes régions) l'intérêt en pourcentage pour un mot clef donné. Les données sont récoltées sur un intervalle de temps, à notre choix.

Figure 5 : Logo de Google

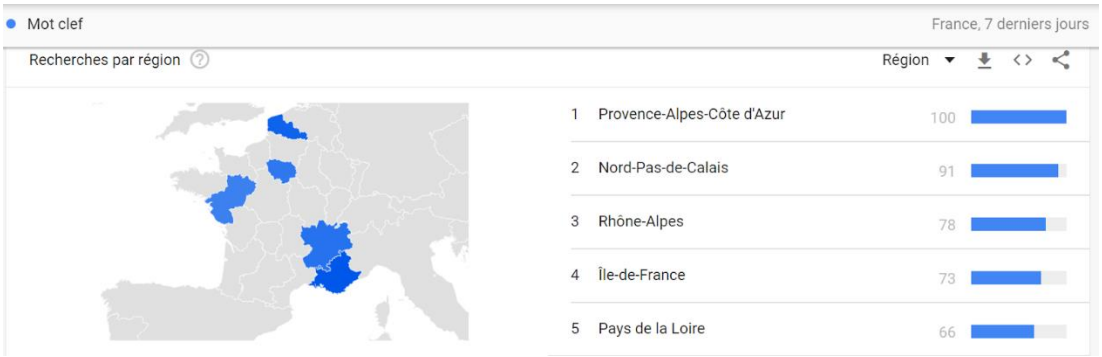


Figure 6 : Capture d'écran de la page web de Google Trends

En revanche, certaines données ne sont pas “initialisées”, c'est-à-dire qu’il n’y a pas de données sur certaines régions ou des mots clés.

Pour ce projet, nous nous baserons sur un pseudo API non officielle de Google Trend avec la bibliothèque PyTrends avec exactement les fonctionnalités que sur le web, nous pourrons par exemple choisir l’intervalle de temps, la localisation, la langue.

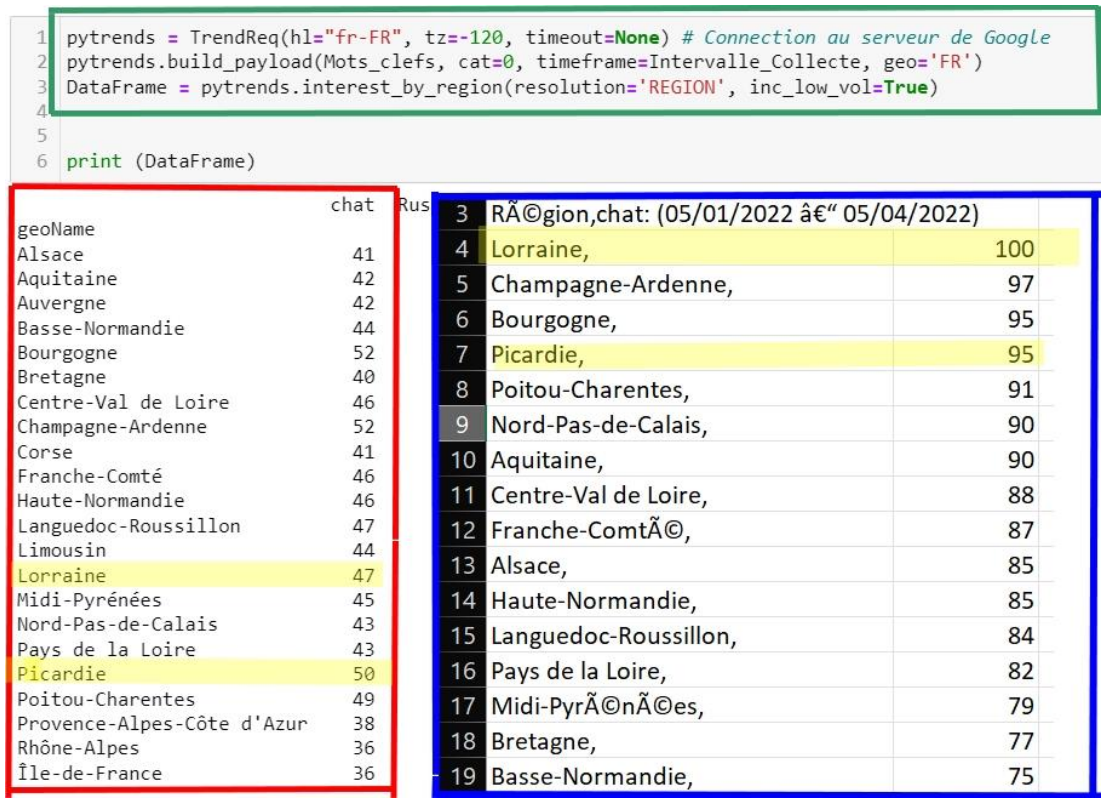
Pour nous un API sera l’interface qui nous permettra d'accéder à Google Trends et de récupérer les données sans passer par une page Web.

Afin d’utiliser cette bibliothèque, nous nous connectons à la base de données de Google, et nous lui faisons des requêtes. Google nous limite à 1300 requêtes par

jour (sur une adresse IP) et limite également les requêtes avec cet API de cinq mots clefs.

L'API nous offre la possibilité de récupérer directement la tendance moyenne sur une période pour toutes les régions d'un pays avec la commande : "pytrends.interest_by_region()".

Nous avons testé l'efficacité de la commande (en vert) :



```
pytrends = TrendReq(hl="fr-FR", tz=-120, timeout=None) # Connection au serveur de Google
pytrends.build_payload(Mots_clefs, cat=0, timeframe=Intervalle_Collecte, geo='FR')
DataFrame = pytrends.interest_by_region(resolution='REGION', inc_low_vol=True)

print (DataFrame)
```

geoName	chat	Rus
Alsace	41	
Aquitaine	42	
Auvergne	42	
Basse-Normandie	44	
Bourgogne	52	
Bretagne	40	
Centre-Val de Loire	46	
Champagne-Ardenne	52	
Corse	41	
Franche-Comté	46	
Haute-Normandie	46	
Languedoc-Roussillon	47	
Limousin	44	
Lorraine	47	
Midi-Pyrénées	45	
Nord-Pas-de-Calais	43	
Pays de la Loire	43	
Picardie	50	
Poitou-Charentes	49	
Provence-Alpes-Côte d'Azur	38	
Rhône-Alpes	36	
Île-de-France	36	

3	RÃ©gion,chat: (05/01/2022 â€“ 05/04/2022)	
4	Lorraine,	100
5	Champagne-Ardenne,	97
6	Bourgogne,	95
7	Picardie,	95
8	Poitou-Charentes,	91
9	Nord-Pas-de-Calais,	90
10	Aquitaine,	90
11	Centre-Val de Loire,	88
12	Franche-ComtÃ©,	87
13	Alsace,	85
14	Haute-Normandie,	85
15	Languedoc-Roussillon,	84
16	Pays de la Loire,	82
17	Midi-PyrÃ©nÃ©es,	79
18	Bretagne,	77
19	Basse-Normandie,	75

Figure 7 : Comparaison des intérêts moyens par région

Nous avons obtenu, en rouge, les données suites à cette commande en vert, puis en bleu nous avons récupéré ces données qui sont censées être identiques sur le site de google trends en bleu sur l'image.

Si nous comparons sur cette image les données pour une même région, le pourcentage d'intérêt varie énormément, sur les lignes jaunes il varie jusqu'à 45%.

Nous avons donc fait ce test sur 5 mots clés différents et cherché un écart moyen par mots-clefs. En moyenne nous obtenons un écart de 31,2% d'intérêt par mots-clefs. Nous avons donc abandonné cette commande.

La deuxième méthode pour récupérer l'intérêt par régions que nous avons adoptée est de construire notre schéma de requête comme l'algorithme suivant :

Soit 5 le nombre de mots clés Maximum pour l'API de google et **T** le nombre de valeurs prise dans un intervalle de temps, **Nb_Gde_R** le nombre de grandes régions et **Nb_Pti_R** le nombre de petites régions.

```
Fonction Recup_Mot_Clef ( Entrée / Liste_Mot_Clef : Tableau de mots clefs allant de 1 à 5,
                          Sortie / Matrice_Data : Matrice de Réels allant de 1 à 5 colonne et 1 à Nb_Gde_R lignes)

Variables :

Index_Grande_Région : Entier
Index_Petite_Région : Entier
Liste_Grande_Région : Tableau de Entier allant de 1 à Nb_Gde_R
Liste_Petite_Région :Tableau de Chaine de caractère allant de 1 à Nb_Pti_R
Temp1 :  Matrice de Réels allant de 1 à 5 colonne et 1 à T lignes

Début :

Initialisation* (Liste_Grande_Région , Liste_Petite_Région)
Index_Grande_Région ← 1
Index_Petite_Région ← 1
Connection_Serveur_google ()

Tant que  ( Index_Grande_Région <= Nb_Gde_R ) Faire

    Temp1 ← 0

    Tant que ( Index_Petite_Région <= Liste_Grande_Région [ Index_Grande_Région ] ) Faire

        Temp1 ← Temp1 + Recup_Trends_API** ( Liste Mot Clé,
                        Liste_Petite_Région [ Index_Petite_Région ] )

        Temp1 ← Temp1 / Liste_Grande_Région [ Index_Grande_Région ]

        Index_Petite_Région ← Index_Petite_Région + 1

    Fin tant que

    Matrice_Data [ Index_Grande_Région  ] ← Moyenne_Par_Colonne ( Temp1 )
    Index_Grande_Région ← Index_Grande_Région  + 1

Fin tant que

Retour : Matrice_Data

Fin
```

Initialisation : Sous-Fonction pour initialiser les listes de grande et petite région ; dans la liste de grande région on obtient un tableau d'entiers, soit pour une grande région plusieurs sous petites régions. Dans la liste des petites régions, on obtient les composantes respectives (de la liste des grandes régions) en chaîne de caractères pour pouvoir les passer en paramètre de Recup_Trends_API.*
*Recup_Trends_API** : Sous-Fonction permettant de récupérer l'intérêt de recherche pour des mots-clefs donnés par rapport à une petite région.*

Figure 8 : Algorithme de récupération des tendances des mots clés par article

La problématique au cœur de cet algorithme a été de trouver une solution efficace pour récupérer les données par anciennes régions et les regrouper avec les nouvelles régions. Nous avons basé cet algorithme pour des couples de 5 mots-clefs d'articles. Car Google ne peut pas prendre plus de 5 mots clés par requête.

Pourquoi regrouper les données par nouvelles régions ? Les chiffres que nous avons en termes de population sont basés sur les nouvelles régions, donc afin d'utiliser les données les plus pertinentes possible il faut regrouper les données sous forme d'une moyenne d'un intervalle de temps par grandes régions.

	Recensement 1990	Recensement 1999	1er janvier 2008	1er janvier 2022 (p)
Auvergne-Rhône-Alpes	6 668 168	6 949 608	7 459 092	8 153 233
Bourgogne- Franche-Comté	2 705 826	2 728 086	2 802 519	2 785 393
Bretagne	2 794 317	2 904 075	3 149 701	3 402 932
Centre - Val de Loire	2 369 808	2 440 295	2 531 588	2 564 915
Corse	249 645	260 152	302 966	349 465
Grand Est	5 274 064	5 387 509	5 521 452	5 542 094
Hauts-de-France	5 770 671	5 855 448	5 931 091	5 987 172
Île-de-France	10 644 665	10 946 012	11 659 260	12 395 148
Normandie	3 126 859	3 202 449	3 293 092	3 307 286
Nouvelle Aquitaine	5 114 287	5 257 954	5 671 076	6 081 985
Occitanie	4 546 249	4 842 680	5 419 946	6 053 548
Pays de la Loire	3 055 197	3 219 960	3 510 170	3 873 096
Provence-Alpes-Côte d'Azur	4 257 244	4 502 385	4 882 913	5 131 187
France métropolitaine	56 577 000	58 496 613	62 134 866	65 627 454

Figure 9 : Recensement officiel de la population au 1er janvier 2022 sur l'INED

Le nombre de requêtes associé augmente en fonction du nombre de mots-clefs que l'on récupère par article. Google limite à 1300 requêtes par jour. Cette limitation nous a posé un problème, car avec cette méthode nous faisons 22 (1 par petite région) requêtes tous les 5 Mots clefs. Cela limite considérablement le nombre d'articles que l'on peut mettre et le nombre de mots clefs que l'on associe à un article.

En revanche, nous avons exploré quelques parties de solutions pour pallier ce problème, notamment instauré un (ou plusieurs) proxy, une solution qui a été vite avortée, car cela impliquerait de mettre en place des serveurs et toutes les 1300

requêtes il serait d'usage de changer de proxy, donc nous avons cherché une solution plus confortable.

La solution qui a été retenue est de mettre un "TimeOut", c'est-à-dire mettre un délai entre chaque requête. Mais cela a pour conséquence directe de rallonger la durée de traitement. Par exemple un TimeOut de 1s, avec 20 articles et 10 mots clefs par articles donnerait 15mins de traitement supplémentaire. Donc nous laissons la possibilité à l'utilisateur de choisir si oui ou non il veut mettre un TimeOut malgré le fait qu'il y est une restriction sur le nombre de requêtes.

4 Analyse des données

Au bout de la chaîne, l'analyse des données. Nous avons en sortie de l'*algorithme 1,* une matrice avec une colonne par mots clefs et une ligne un % d'intérêt par grande région. Il nous suffit donc de regrouper les données en volume de requêtes par régions et sur le total de la France. Le volume de requêtes par région n'est pas une donnée que l'utilisateur pourra récupérer. Nous appliquons la méthode suivante :

Liste des grandes régions	Intéret (%)	Population de la grande Région	Volume de requêtes *(Intéret * Pop Gde Région / Total Pop)*
Centre-Val-de-Loire	**99%**	**2 564 915**	**0,04**
Île-de-France	**50%**	**12 395 148**	**0,09**
...	...	...	...
Pays de la Loire	**69%**	**3 873 096**	**0,04**
Provence-Alpes-Côte d'Azur	**30%**	**5 131 187**	**0,02**
Total de la population :	**65 000 000**		
Volume de requête en France :	**0,04 + 0,09 + ... + 0,04 + 0,02 = 0,57**		

Figure 10 : Récupération du volume de requêtes

Cette méthode est appliquée pour chaque mot clef. Nous récupérons un volume de requêtes en France par mots clefs.

Concernant des requêtes qui ne sont pas initialisées sur certaines régions, on a donc une moyenne de l'intérêt sur un intervalle de temps de 0. Sur ces valeurs on retranche à l'effectif total. Cela nous donnera un volume de requêtes approchant la réalité.

Ensuite, nous avons trié les données afin de mettre les mots clefs avec le volume de requête le plus élevé tout en haut de la liste du fichier final. Nous avons utilisé un simple algorithme de Tri par sélection.

```
procédure tri_selection(tableau t)
    n ← longueur(t)
    pour i de 0 à n - 2
        min ← i
        pour j de i + 1 à n - 1
            si t[j] < t[min], alors min ← j
        fin pour
        si min ≠ i, alors échanger t[i] et t[min]
    fin pour
fin procédure
```

Figure 11 : Tri par sélection de Wikipédia

Nous obtenons une liste de mots clefs triés comme l'exemple fait de toute pièce ci-dessous :

	A	B
1	Mot-Clef	Volume de requetes en france
2	TEST5	0.9
3	TEST7	0.9
4	TEST8	0.8798
5	TEST6	0.8
6	TEST10	0.78938314
7	FINTEST11	0.2342347
8	TEST9	0.18348329
9	TEST4	0.14
10	TEST3	0.11
11	TEST1	0.1
12	TEST2	0.01
13		

Figure 12 : Exemple de fichier final trié

Pour le Fichier final nous avons mis la date dans le nom afin qu'il soit unique exemple ci-dessous :

Figure 13 : Exemple de nom du fichier final

Le fichier final est fait à partir de la matrice triée, et mis sous forme de csv, avec les colonnes Mot-Clefs et Volume de requête en France.

5 Gestion de projet

Dans un premier temps, pour la globalité du projet, un programme en python nous a paru évident, car nous n'avons pas besoin de gérer du code bas niveau et il existe plus de bibliothèques appropriées au web en python (comme PyTrends que l'on ne trouve pas forcément dans d'autres langages). Le langage python n'est également pas compliqué à comprendre et facile à l'apprentissage.

Nous avons réparti le projet en plusieurs "tâches principales" comme le diagramme de Gantt suivant :

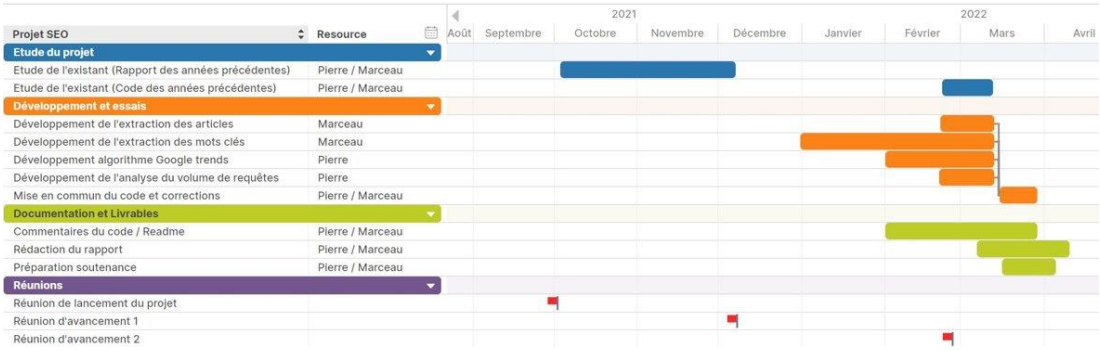


Figure 14 : Diagramme de Gantt

(* Diagramme en grand format dans les annexes)

Nous avons eu beaucoup de mal à saisir l'étendue du projet, nous nous sommes dirigés à l'issue de la première réunion vers la création d'un site web. Au fur et à mesure des réunions nous nous sommes rendu compte que nous étions hors sujet. Pour nous aiguiller au mieux, Mr Delalandre nous a donné les livrables des années précédentes, ce qui pour la compréhension du sujet nous a bien aidés.

À la suite de cela nous avons réparti intelligemment les tâches avec des sous-tâches plus ou moins complexes, puis nous avons avancé petit à petit sur notre programme.

Nous avons obtenu par la suite une partie des programmes des années précédentes, mais certains points ne convenaient pas à nos besoins ou n'étaient pas compréhensibles facilement. Nous nous sommes uniquement appuyés sur ces programmes pour fournir un programme original et compréhensible en ajoutant des commentaires pertinents et un “ReadMe” complet.

6 Conclusion

Ce projet a été pour nous très formateur et intéressant. Nous avons eu l'occasion d'approfondir notre programmation en python. L'aspect SEO du projet nous a également permis d'utiliser diverses bibliothèques afin d'approfondir et d'exploiter au mieux l'extraction des mots clefs ou encore d'utiliser des API's pour récupérer les données transmises par Google Trends.

De plus, nos recherches pour réaliser ce projet nous ont permis d'apprendre énormément sur la notion de SEO ainsi que sur la thématique de la data science en général. Le fait d'avoir énormément de ressources sur le web nous a grandement aidé pour acquérir bon nombre de connaissances.

Il reste néanmoins à nos yeux quelques axes d'améliorations comme la suppression de phrases clés se ressemblant et l'analyse des données sur les variations. En effet, concernant l'extraction des données via PyTrends, il serait possible d'analyser clairement les variations temporelles plutôt que de faire une simple moyenne, et d'en déduire un résultat peut-être plus pertinent.

Ces axes d'améliorations se verront exploités par la suite dans une éventuelle reprise du projet. Nous sommes toutefois très satisfaits du travail que nous avons réalisé et nous avons apprécié réaliser le projet développement sur cette thématique.

Tables des illustrations

Annexes

Annexe 1 : Datasheet – KeyBERT (Paramètres)

Name	Type	Description	Default
`docs`	`Union[str, List[str]]`	The document(s) for which to extract keywords/keyphrases	*required*
`candidates`	`List[str]`	Candidate keywords/keyphrases to use instead of extracting them from the document(s)	`None`
`keyphrase_ngram_range`	`Tuple[int, int]`	Length, in words, of the extracted keywords/keyphrases	`(1, 1)`
`stop_words`	`Union[str, List[str]]`	Stopwords to remove from the document	`'english'`
`top_n`	`int`	Return the top n keywords/keyphrases	`5`
`min_df`	`int`	Minimum document frequency of a word across all documents if keywords for multiple documents need to be extracted	`1`
`use_maxsum`	`bool`	Whether to use Max Sum Similarity for the selection of keywords/keyphrases	`False`
`use_mmr`	`bool`	Whether to use Maximal Marginal Relevance (MMR) for the selection of keywords/keyphrases	`False`
`diversity`	`float`	The diversity of the results between 0 and 1 if use_mmr is set to True	`0.5`
`nr_candidates`	`int`	The number of candidates to consider if use_maxsum is set to True	`20`
`vectorizer`	`CountVectorizer`	Pass in your own CountVectorizer from scikit-learn	`None`
`highlight`	`bool`	Whether to print the document and highlight its keywords/keyphrases. NOTE: This does not work if multiple documents are passed.	`False`
`seed_keywords`	`List[str]`	Seed keywords that may guide the extraction of keywords by steering the similarities towards the seeded keywords	`None`

Annexe 2 : Datasheet – Newspaper3k (GitHub)

```
>>> from newspaper import Article

>>> url = 'http://fox13now.com/2013/12/30/new-year-new-laws-obamacare-pot-guns-and-drones/'
>>> article = Article(url)
```

```
>>> article.download()

>>> article.html
'<!DOCTYPE HTML><html itemscope itemtype="http://...'
```

```
>>> article.parse()

>>> article.authors
['Leigh Ann Caldwell', 'John Honway']

>>> article.publish_date
datetime.datetime(2013, 12, 30, 0, 0)

>>> article.text
'Washington (CNN) -- Not everyone subscribes to a New Year's resolution...'

>>> article.top_image
'http://someCDN.com/blah/blah/blah/file.png'

>>> article.movies
['http://youtube.com/path/to/link.com', ...]
```

Annexe 3 : Code ISO 3166-2 des régions de France

ZEAT (NUTS de niveau 1)			Région (NUTS de niveau 2)			
Codification		Nom NUTS-1[127, 128]	Codification			Nom de la région[74] (avant 2016)
NUTS-1 (2013)[128]	Insee[74]		ISO 3166-2[129]	NUTS-2 (2013)[128]	Insee[74]	
FR1	1	Île-de-France	FR-J	FR10	11	Île-de-France
FR2	2	Bassin parisien	FR-G	FR21	21	Champagne-Ardenne
			FR-S	FR22	22	Picardie
			FR-Q	FR23	23	Haute-Normandie
			FR-F	FR24	24	Centre-Val de Loire
			FR-P	FR25	25	Basse-Normandie
			FR-D	FR26	26	Bourgogne
FR3	3	Nord - Pas-de-Calais	FR-O	FR30	31	Nord-Pas-de-Calais
FR4	4	Est	FR-M	FR41	41	Lorraine
			FR-A	FR42	42	Alsace
			FR-I	FR43	43	Franche-Comté
FR5	5	Ouest	FR-R	FR51	52	Pays de la Loire
			FR-E	FR52	53	Bretagne
			FR-T	FR53	54	Poitou-Charentes
FR6	7	Sud-Ouest	FR-B	FR61	72	Aquitaine
			FR-N	FR62	73	Midi-Pyrénées
			FR-L	FR63	74	Limousin
FR7	8	Centre-Est	FR-V	FR71	82	Rhône-Alpes
			FR-C	FR72	83	Auvergne
FR8	9	Méditerranée	FR-K	FR81	91	Languedoc-Roussillon
			FR-U	FR82	93	Provence-Alpes-Côte d'Azur
			FR-H	FR83	94	Corse
FRA	0	Département d'outre-mer	FR-GP	FRA1	01	Guadeloupe
			FR-MQ	FRA2	02	Martinique
			FR-GF	FRA3	03	Guyane
			FR-RE	FRA4	04	La Réunion
			FR-YT	FRA5	06	Mayotte

Annexe 4 : Diagramme de Gantt