Compte rendu test pratique stage été 2025

Auteur

MAKALOU Shérif

Apprenti ingénieur en développement et cybersécurité, 5ème année

INSA Centre-Val-de-Loire, Bourges, FRANCE

Année Universitaire 2024 - 2025

version : 31 mars 2025

Table des matières

1 Contexte / Introduction

Ce rapport fait suite à un test pratique qui a été réalisé dans le cadre du processus de sélection pour un stage d'été en développement à L'ÉTS de Montréal. Il se décompose en deux parties : une partie **automatisation/extraction de données**, et une partie d'**étude de faisabilité**. Ce rapport présentera donc le travail que j'ai réalisé en montrant mon approche sur chacune des tâches, les outils utilisés, quelques captures des résultats obtenus, mais aussi une analyse critique.

2 (Partie 1) - Automatisation, extraction de données, création de graphiques

But de l'exercice : Identifier les contributions en collaboration entre deux entités et extraire des données pertinentes.

* Regrouper les publications attribuables à l'ÉTS (https ://ror.org/0020snb74) pour la période 2019 à 2023. https ://docs.openalex.org/api-entities/institutions/institution-object

* Programmer l'extraction de la liste des pays collaborateurs pour la période.

* Réaliser un graphique présentant les 10 principaux pays collaborateurs à insérer dans un document Word.

Il y avait de plus, une tâche à réaliser parmi 4 autres. J'ai choisi la tâche numéro 2 : *Programmer l'extraction de la liste des principaux sujets des publications en collaboration entre l'École de technologie supérieure, à Montréal, et le Centre National de la Recherche Scientifique (CNRS) (https ://ror.org/02feahw73), en France. Réaliser un graphique présentant cette liste de sujets.*

2.1 Outils/Technologies utilisés

J'ai naturellement décidé d'utiliser le langage de programmation Python pour plusieurs raisons : c'est le langage le plus célèbre quand il s'agit de faire de l'automatisation et du traitement de données, c'est le langage qui est apparemment utilisé dans le projet autoBib+, et c'est aussi le langage avec lequel je suis le plus à l'aise en programmation.

En fonction des besoins, j'ai fait appel à plusieurs bibliothèques (Pandas, Matplotlib, Tkinter, etc.) pour répondre aux différentes tâches. Étant donné que j'ignore la nature des environnements (présence ou absence de Python et des bibliothèques) sur lesquels mon code sera testé, j'ai décidé de générer un fichier exécutable **(.exe)** qui sera accompagné des dll Python nécessaires au bon fonctionnement du programme. L'exécutable a été généré à l'aide de **cx-Freeze**.

2.2 Reformulation du problème et approche

Il est important avant tout d'expliquer le contexte dans lequel j'ai effectué le travail **i.e.** la manière dont j'ai compris l'énoncé. Il est bien possible, dans le pire des cas, que le travail demandé ne corresponde pas à celui réalisé si jamais je n'ai pas compris les consignes comme il le fallait. Dans le cas échéant, il est important d'au moins noter une cohérence entre ce que je pense avoir compris et ce que j'ai réalisé. Je vais donc poser les bases en expliquant comment j'ai considéré les choses.

Dans un premier temps, j'ai dû me renseigner un peu sur le fonctionnement de l'api OpenAlex avec la documentation fournie. J'ai pu ainsi comprendre qu'OpenAlex était un grand catalogue d'articles scientifiques, d'auteurs et d'institutions qui dispose notamment d'une api permettant d'extraire des tonnes d'informations sous format JSON avec plein de dictionnaires d'attributs permettant de traiter efficacement les données qui nous intéressent. Après avoir saisi le fonctionnement de l'api j'ai pu avoir une idée sur la manière de traiter l'exercice.

Ainsi, voici comment j'ai considéré les choses :

- On considère qu'une publication est attribuable à l'ÉTS si au moins un des auteurs de la publication est associé à l'ÉTS (attribut **authorships.institutions.ror**).
- Un pays est considéré comme collaborateur avec l'ÉTS sur un article donné si au moins une institution associée à un des auteurs de l'article se trouve dans le pays en question (attribut **authorships.institutions.country-code**).
- Les pays collaborateurs principaux de l'ÉTS sont les pays ayant le plus grand nombre d'auteurs qui ont participé à la rédaction des articles attribuables à l'ÉTS.

2.3 Programmation et exécution

Pour implémenter ma solution, je suis parti sur une **approche modulaire** avec de la **programmation orientée objet** afin d'obtenir un code lisible et facilement modifiable. Mon programme se divise en deux fichiers :

- **classes.py** qui va être le fichier dans lequel se trouveront les méthodes principales qui se chargeront de l'analyse, l'extraction des données et de la génération des graphiques.
- **gui.py** qui va être mon fichier où sera implémenté l'interface, ce sera le point d'entrée pour l'exécution du programme. J'ai utilisé **Tkinter** pour réaliser une interface utilisateur.

Chacune des méthodes implémentées dans les différents fichiers est commentée afin de comprendre les opérations qu'elle effectue. Je vais maintenant montrer comment interagir avec l'interface et montrer les résultats attendus.

NB : Les mesures de sécurité mises en place par la Direction des Systèmes d'Informations de votre établissement pourraient empêcher le bon fonctionnement de l'exécutable car les fichiers **.exe** non signés sont généralement considérés comme potentiellement malveillants. Si vous êtes dans cette situtation, il est préférable alors d'exécuter directement le fichier python **gui.py** dans un environnement approprié (avec Python d'installé ainsi que les bibliothèques nécessaires), ce qui est plus contraignant.

Exécution du programme

Lorsque vous lancerez l'exécutable, vous verrez l'interface suivante générée à l'aide de Tkinter :

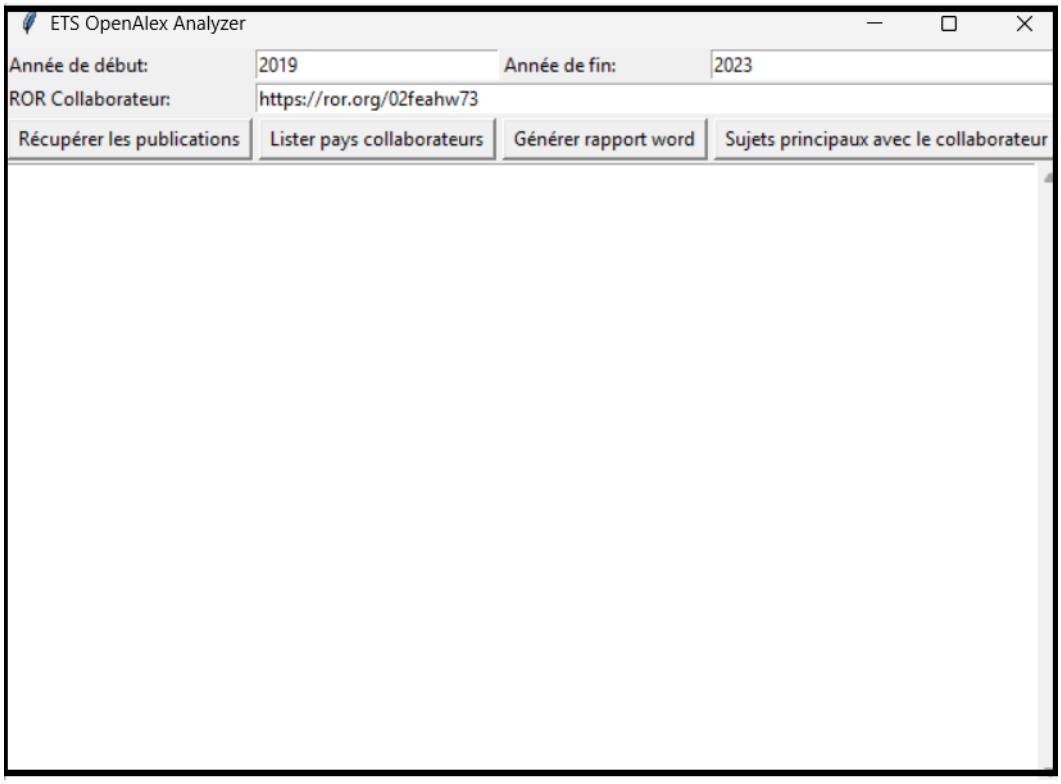

FIGURE 1 – Interface du programme

On a 4 boutons qui correspondent aux 4 consignes qui ont été données dans l'exercice. Les champs **Année de début** et **Année de fin** sont globaux à tous les boutons, ils ne peuvent contenir que des nombres, et les années sont obligatoirement comprises entre les années 1981(année à partir de laquelle on arrive à extraire des informations via OpenAlex) et 2025. Le champ **ROR collaborateur** ne concerne que le bouton **"Sujets principaux avec le collaborateur"**. Si un ROR est invalide, l'interface nous le notifiera.

Voici une description de chacun des boutons :

- **Récupérer les publications** permet de regrouper les publications attribuables à l'ÉTS sur la période définie par les champs des années. Lorsque vous cliquerez dessus, vous aurez un petit temps de chargement et des informations s'afficheront dans l'interface :

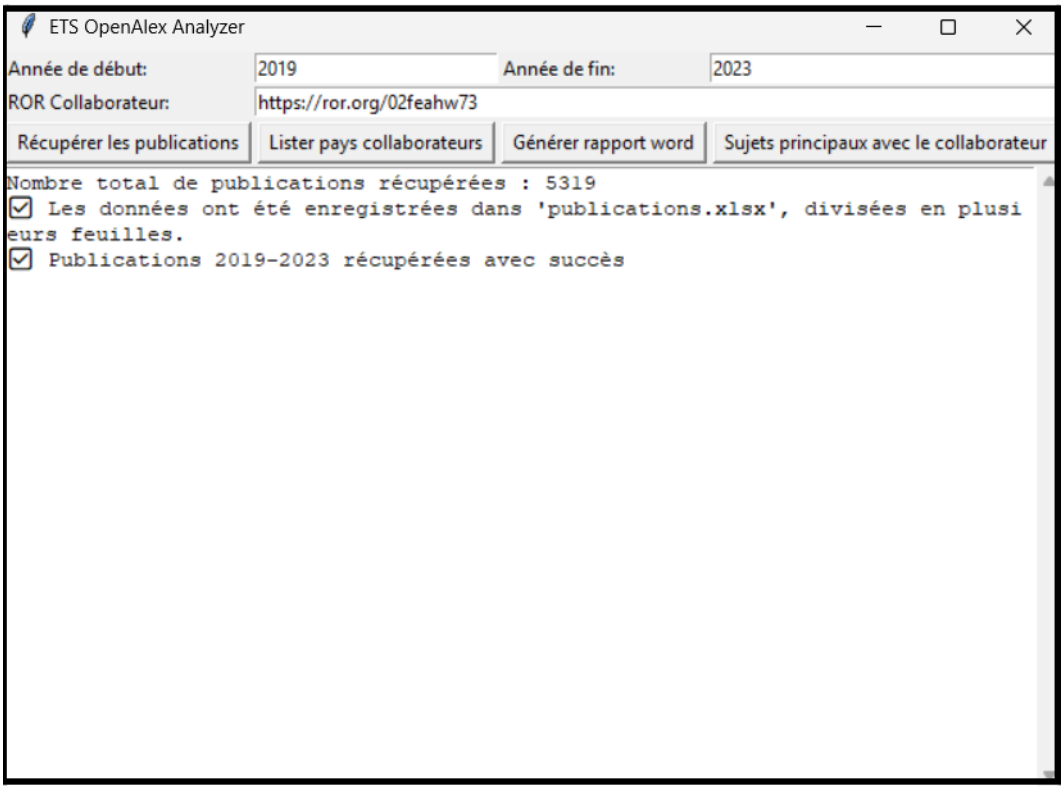

FIGURE 2 – Résultat de récupération des publications

Comme indiqué, les résulats se trouvent dans un fichier Excel nommé **publications.xlsx** qui aura plusieurs **sheets** selon la taille des données récupérées. Ce fichier se trouvera dans le même répertoire que l'exécutable gui.exe. Voici un aperçu :

	Titre	Année	Lien vers l'article
2	On the Susceptibility of SRAM-Based FPGA Routing Network to Delay Ch	2019	https://doi.org/10.1109/tns.2019.2898894
3	A Mathematical Model: A Flexible Manufacturing System, Prone to Erroı	2019	https://doi.org/10.4236/ajibm.2019.91011
4	The ClimEx Project: A 50-Member Ensemble of Climate Change Projectic	2019	https://doi.org/10.1175/jamc-d-18-0021.1
5	Intelligent Resource Management Based on Reinforcement Learning for	2019	https://doi.org/10.1109/tvt.2018.2890686
6	Software Development Effort Estimation Using Regression Fuzzy Models	2019	https://doi.org/10.1155/2019/8367214
7	Benchmark on Automatic Six-Month-Old Infant Brain Segmentation Algc	2019	https://doi.org/10.1109/tmi.2019.2901712
8	UAV Trajectory Planning for Data Collection from Time-Constrained IoT	2019	https://doi.org/10.1109/twc.2019.2940447
9	Recent progress in the preparation, properties and applications of super	2019	https://doi.org/10.1016/j.porgcoat.2019.03.042
10	IMPACT World+: a globally regionalized life cycle impact assessment me	2019	https://doi.org/10.1007/s11367-019-01583-0
11	SACO: A Service Chain Aware SDN Controller-Switch Mapping Framewor	2019	https://doi.org/10.23919/cnsm46954.2019.9012747
12	Design optimization of aero-engine turbine blade and disc fixing	2019	https://doi.org/10.15406/aaoaj.2019.03.00086
13	A Deep Neural Network based Approach to Energy Efficiency Analysis fc	2019	https://doi.org/10.1109/indin41052.2019.8972019
14	Reconfigurable IR-UWB Current Mode Switched Receiver for IoT Applica	2019	https://doi.org/10.1109/icecs46596.2019.8965095
15	On the performance of the maximum likelihood estimation method for	2019	https://doi.org/10.3850/978-981-11-2724-3_0888-cd
16	A new fast compensator design applied to a quadcopter	2019	https://doi.org/10.1109/icsc47195.2019.8950601
17	Multi-PVT-Point Analysis and Comparison of Recent Small-Delay Defect	2019	https://doi.org/10.1007/s10836-019-05832-w
18	Adaptation of person re-identification models for on-boarding new cam	2019	https://doi.org/10.1016/j.patcog.2019.106991
19	Strength of pozzolan soil blend in chemically improved lateritic soil for p	2019	https://doi.org/10.1093/ijlct/ctz035
20	White matter fiber analysis using kernel dictionary learning and sparsity	2019	https://doi.org/10.1016/j.patcog.2019.06.002
21	Power-Efficient Video Streaming on Mobile Devices Using Optimal Spati	2019	https://doi.org/10.1109/icce-berlin47944.2019.896617
22	Valorization and sequestration of hydrogen gas from biomass combusti	2019	https://doi.org/10.1016/j.mset.2019.11.003
23	Influence of Nickel on High-Temperature Oxidation and Characteristics	2019	https://doi.org/10.1002/srin.201900536
24	The Matchstick Model for Anisotropic Friction Cones	2019	https://doi.org/10.1111/cgf.13885
25	Pybotics: Python Toolbox for Robotics	2019	https://doi.org/10.21105/joss.01738
26	Commercial Aircraft Trajectory Optimization to Reduce Flight Costs and	2019	https://doi.org/10.1007/978-981-13-9806-3_2
27	Container Based Resource Management for Data Processing on IoT Gat	2019	https://doi.org/10.1016/j.procs.2019.08.034

Sheet_1 Sheet_2 Sheet_3 Sheet_4 Sheet_5 Sheet_6 +

FIGURE 3 – Fichier Excel contenant les publications

J'ai décidé de n'afficher que 3 informations concernant les publications : le titre, l'année de publication et le lien vers l'article. Les articles sont triés par année.

- **Lister pays collaborateurs** permet l'extraction de la liste des pays collaborateurs pour la période précisée. En cliquant sur ce bouton, vous allez générer un fichier Excel nommé **Pays-collaborateurs-ets.xlsx** qui se trouvera aussi au même niveau que le fichier .exe. Ce fichier contiendra la liste des pays collaborateurs ainsi que le nombre de publications attribuables à chaque pays.

	A	B
1	**Pays**	**Nombre de publications**
2	CA	18947
3	US	2118
4	FR	1805
5	CN	1284
6	GB	725
7	IR	556
8	DE	539
9	IN	482
10	BR	478
11	IT	386
12	JP	383

FIGURE 4 – Fichier Excel contenant les pays collaborateurs de l'ÉTS

- **Générer rapport word** permet de réaliser un graphique présentant les 10 principaux pays collaborateurs et insère le tout dans un document word nommé **rapport-collaborations.docx**.

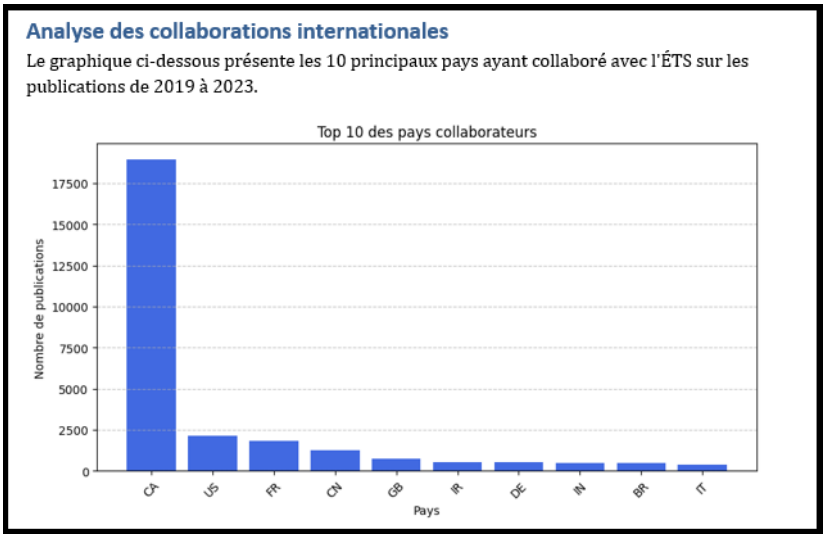

FIGURE 5 – Aperçu du document word généré

- **Sujets principaux avec le collaborateur** permet d'extraire la liste des principaux sujets des publications entre l'École de Technologie Supérieure, à Montréal(ÉTS) et un institut dont le ROR sera précisé dans le champ **ROR collaborateur** sur la période précisée. Par défaut, c'est le ROR du Centre National de la Recherche Scientifique (CNRS) en France qui a été renseigné. Ce bouton va générer une image **top-topics.png** qui montrera une diagramme identique à celui des collaborations internationales dans lequel les principaux sujets seront représentés (top 20).

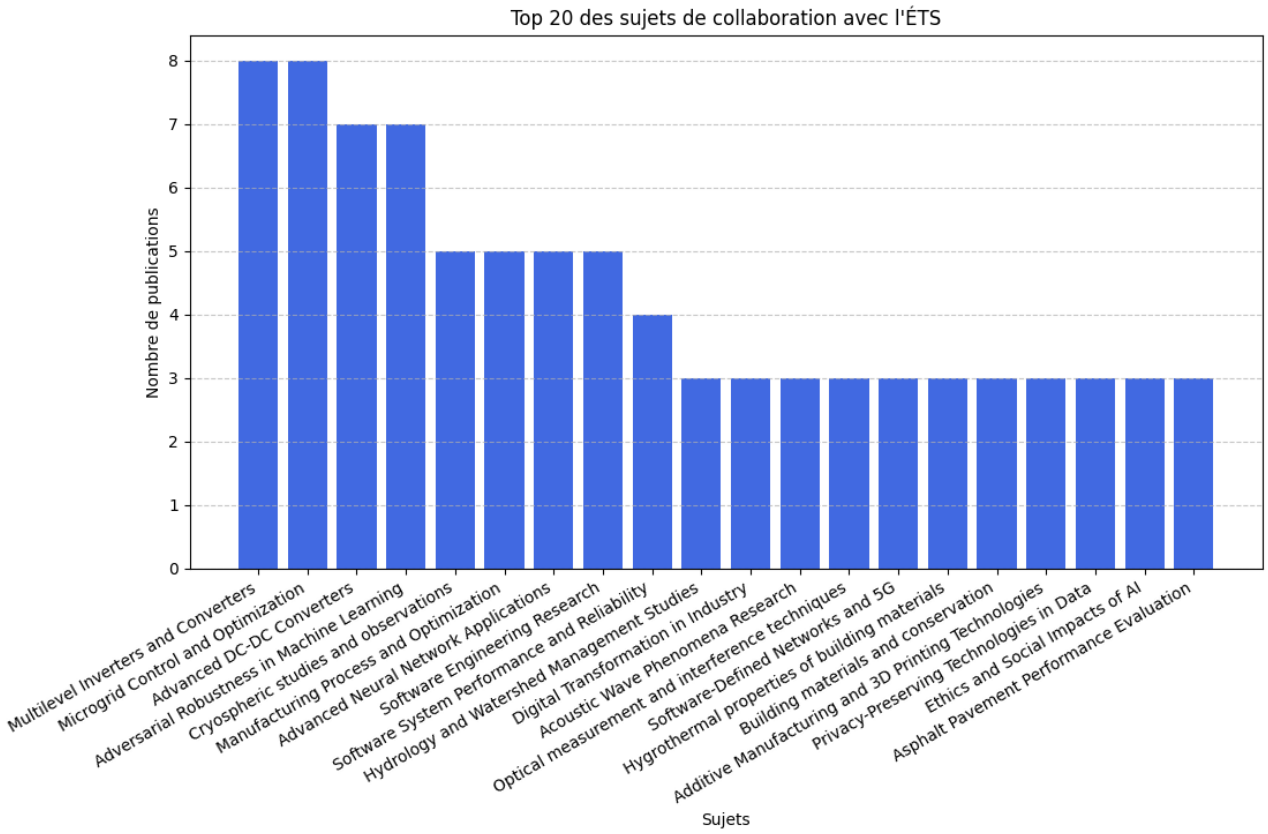

FIGURE 6 – Principaux sujets de collaborations

Attention : avant toute nouvelle tentative d'extraction de données, veillez toujours à fermer les anciens fichiers Excel/Word ouverts. Si un fichier reste ouvert, son accès en écriture sera restreint, et aucune génération ne se fera.

3 (Partie 2) - Rapport de faisabilité

Dans le cadre d'une initiative visant à optimiser l'analyse des profils de recherche, les équipes souhaitent générer automatiquement un tableau récapitulatif des **SciVal Topics** (grappes thématiques) associés aux publications d'un chercheur, tel que présenté dans l'exemple joint. Ce tableau permet d'identifier les domaines de spécialisation d'un chercheur en regroupant ses publications par sujets connexes (exemple : « Microplastics, Water Pollutant, Environmental Monitoring »). L'objectif de ce rapport est d'évaluer la possibilité d'automatiser cette tâche à l'aide des **APIs Elsevier** et de proposer des solutions alternatives le cas échéant.

3.1 Analyse des outils et choix de l'API

Après analyse de la documentation, je pense que l'API la plus adaptée est celle de **Scopus**. Dans la suite, je vais expliquer mon choix ainsi que les contraintes identifiées.

Scopus est une base de données bibliographique de référence contenant les métadonnées de millions de publications scientifiques (titres, auteurs, résumés, mots-clés, etc.). Son API permet d'interroger ces données de manière programmatique. Par exemple, il est possible de récupérer l'ensemble des publications d'un chercheur via son identifiant Scopus (**dc :identifier**), ainsi que les mots-clés associés à chaque publication (**authkeywords**). Ces mots-clés, bien que non standardisés, constituent la base nécessaire pour identifier des thématiques récurrentes. Cependant, Scopus ne fournit pas directement les **SciVal Topics**, qui sont des clusters thématiques générés par un algorithme propriétaire d'Elsevier. Ces clusters, utilisés dans la plateforme SciVal, regroupent des publications similaires en fonction de critères complexes (mots-clés, citations, co-occurrences de termes). L'absence d'API publique pour SciVal pour la récupération de telles informations rend impossible l'accès direct à ces données.

En effet, SciVal est un outil d'analyse avancée des données de Scopus, conçu pour évaluer la performance de la recherche (collaborations, tendances thématiques, benchmarking). Les SciVal Topics y sont calculés automatiquement et mis à jour régulièrement. Bien que ces données soient précieuses, leur génération repose sur des modèles propriétaires non documentés, et aucune API n'est disponible pour les exporter. Ainsi, les grappes thématiques présentées dans l'exemple joint à l'énoncé de l'exercice ne peuvent être répliquées fidèlement sans accès manuel à la plateforme SciVal.

3.2 Faisabilité technique de l'automatisation

Scopus dispose d'un certain nombre d'api, voici celles qui vont nous être principalement utiles :

- **Scopus Search** pour trouver des articles à partir de titres, mots-clés, id de l'auteur etc.
- **Author Search** pour identifier un auteur.

- **Author Retrieval** pour obtenir les publications d'un chercheur.

Ainsi nous pouvons d'abord retrouver l'id d'un auteur à l'aide de **Author Search** et du paramètre **dc :identifier**, ainsi que les sujets principaux sur lesquels il travaille grâce au paramètre **subject-area**. Cependant, on dirait qu'on est limité à 3 sujets maximum, ce qui pourrait être contraignant dépendemment du besoin exprimé. (https://dev.elsevier.com/sc_author_search_views.html#:~:text=Subject%20Areas-,(Maximum%20of%203),-X).
De plus, les sujets pourraient être un peu trop généraux et donc, pas aussi spécifiques que les Topic clusters.
Ensuite, à l'aide de l'id de l'auteur récupéré ainsi que de **Author Retrieval**, nous pouvons obtenir les publications d'un chercheur (titre, DOI, Date de publication etc)
Enfin, à l'aide de **Scopus Search**, nous pouvons récupérer les topics associés aux différents articles grâce au paramètre **authkeywords**.

Nous disposons ainsi de la quasi-totalité des informations nécessaires pour faire notre automatisation. Cependant, il nous manque l'association entre les **Topic clusters** et les **topics**. Cela demandera donc un travail supplémentaire qu'il faudra effectuer indépendamment des apis.
En effet, en l'absence d'accès aux SciVal Topics, une approximation peut être réalisée en combinant les données de Scopus avec des techniques de **traitement du langage naturel (NLP)**. Le traitement automatique du langage naturel est un domaine multidisciplinaire impliquant la linguistique, l'informatique et l'intelligence artificielle. Il vise à créer des outils de capable d'interpréter et de synthétiser du texte pour diverses applications. Ainsi, une technique de NLP comme **DBSCAN** ou **K-means** pourrait nous permettre de générer des clusters pertinents à partir d'une liste de mots clés, ce qui nous permettrait d'avoir un tableau complet répondant au besoin des collègues. Il ne faudra par contre, pas s'attendre à ce que les clusters générés soient aussi pertinents que ceux obtenus par Scival.

Voici un exemple de code permettant de générer des clusters. Ici l'algorithme va plutôt analyser les occurrences des mots présents dans la liste. Les noms des clusters sont aussi basés sur les occurrences de mots clés.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
import numpy as np

# 1. Donnees d'exemple
keywords = [
    # Intelligence Artificielle
    "machine learning", "deep learning", "neural networks",
    "transfer learning", "generative adversarial networks", "
       transformers",

    # Sciences Environnementales
    "climate change", "carbon footprint", "sustainable development",
    "renewable energy", "biodiversity loss", "circular economy",
```

```
14
15      # Science des Materiaux
16      "graphene", "carbon nanotubes", "nanocomposites",
17      "metamaterials", "biomaterials", "smart materials",
18
19      # Sante/Biologie
20      "CRISPR", "gene therapy", "vaccine development",
21      "cancer immunotherapy", "stem cells", "tissue engineering",
22      "antibiotic resistance", "microbiome", "precision medicine",
23
24      # Physique/Chimie
25      "quantum computing", "superconductivity", "plasmonics",
26      "catalysis", "batteries", "fuel cells",
27
28       # Sciences Sociales
29      "behavioral economics", "social inequality", "urban planning",
30      "political polarization", "fake news", "algorithmic bias",
31
32  ]
33
34  # 2. Vectorisation
35  vectorizer = TfidfVectorizer(stop_words='english')
36  X = vectorizer.fit_transform(keywords)
37
38  # 3. Clustering
39  n_clusters = min(6, len(keywords))  # Maximum 4 clusters
40  kmeans = KMeans(n_clusters=n_clusters, random_state=42)
41  clusters = kmeans.fit_predict(X)
42
43  # 4. Generation des noms de clusters
44  def generate_cluster_names(terms, labels, n_words=3):
45      unique_labels = np.unique(labels)
46      cluster_names = {}
47
48      for label in unique_labels:
49          # Recupere les indices des termes du cluster
50          indices = np.where(labels == label)[0]
51
52          # Compte les mots dans le cluster
53          word_counts = {}
54          for idx in indices:
55              for word in terms[idx].split():
56                  word_counts[word] = word_counts.get(word, 0) + 1
57
58          # Selectionne les n mots les plus frequents
59          sorted_words = sorted(word_counts.items(), key=lambda x: x[1],
                reverse=True)
60          cluster_name = " ".join([word for word, count in sorted_words[:
                n_words]])
61
62          cluster_names[label] = cluster_name
63
64      return cluster_names
```

```
65
66 # 5. Affichage des resultats
67 cluster_names = generate_cluster_names(keywords, clusters)
68
69 print("CLUSTERS␣THEMATIQUES␣IDENTIFIES:\n")
70 for cluster_id, name in cluster_names.items():
71     cluster_keywords = [kw for kw, cl in zip(keywords, clusters) if cl
       == cluster_id]
72     print(f"Cluster␣'{name}':")
73     print("␣.␣" + "\n␣.␣".join(cluster_keywords) + "\n")
```

Bien évidemment, cet algortithme est très simpliste, le clustering ne sera pas forcément pertinent. Mais au moins ça donne une idée de ce qu'il est possible de faire. L'utilisation d'algorithmes plus puissants et l'usage de l'intelligence artificielle pourraient permettre d'obtenir de bien meilleurs résulats. Voici ce qu'on obtient :

```
1     CLUSTERS THEMATIQUES IDENTIFIES:
2
3 Cluster 'carbon␣transformers␣climate␣change␣footprint':
4      transformers
5      climate change
6      carbon footprint
7      renewable energy
8      biodiversity loss
9      circular economy
10     graphene
11     carbon nanotubes
12     nanocomposites
13     metamaterials
14     biomaterials
15     smart materials
16     CRISPR
17     gene therapy
18     cancer immunotherapy
19     tissue engineering
20     quantum computing
21     superconductivity
22     plasmonics
23     catalysis
24     batteries
25     behavioral economics
26     urban planning
27     political polarization
28     fake news
29     algorithmic bias
30
31 Cluster 'cells␣stem␣fuel':
32     stem cells
33     fuel cells
34
35 Cluster 'learning␣machine␣deep␣transfer':
36     machine learning
```

```
      deep learning
      transfer learning

Cluster 'social inequality':
      social inequality

Cluster 'networks neural generative adversarial':
      neural networks
      generative adversarial networks

Cluster 'development sustainable vaccine':
      sustainable development
      vaccine development
```

3.3 Conslusion de l'étude

Bien que les APIs Elsevier ne permettent pas de répliquer exactement les SciVal Topics, une automatisation partielle est réalisable en exploitant les données de Scopus et des techniques de clustering. Cette approche nécessite un investissement initial en développement et en validation, mais offre une alternative viable pour les institutions sans accès à SciVal. Pour une fidélité maximale aux données d'origine, l'export manuel depuis SciVal demeure la solution recommandée.

4 Analyse critique des travaux

Les travaux effectués répondent peut-être aux besoins du cahier des charges, mais il ne sont cependant pas parfaits. J'ai relevé quelques améliorations qui auraient pu être faites si j'avais disposé d'un peu plus de temps.

Partie 1 : Je trouve dans un premier temps que les requêtes effectuées pour l'extraction des données sont assez lentes. Cela ne me surprend pas car il existe dans le code des méthodes à forte complexité ($O(n^3)$ pour la méthode privée **___extract_collaborators()** par exemple). Et donc, plus la plage d'années sur laquelle on veut récupérer les données est grande, plus le temps de traitement augmente de manière significative. J'avais trouvé une API qui s'appelle **PyAlex** et qui permet d'effectuer des requêtes à OpenAlex pour récupérer des données. Peut-être que cette API aurait permis une extraction plus rapide, mais j'ai décidé de ne pas l'utiliser car j'ignorais son niveau de fiabilité et que je ne voyais pas en quoi il aurait été plus bénéfique de l'utiliser.

D'autre part, l'interface que j'ai proposée est assez vieillissante et aurait pu avoir un design plus moderne. Des outils comme **PyQT** auraient permis d'avoir une interface plus vivante. Une autre possibilité aurait été de mettre en place une application web afin de gérer l'interfaçage de manière plus simple et efficace.

Partie 2 : Mes analyses sont assez théoriques, ce qui est normal vu que c'est une étude et non un travail pratique comme dans la première partie. Mais je pense qu'une utilisation appronfie des différentes api citées et un accès à la plateforme Scival (qui est payant) m'auraient permis d'avoir un regard plus clair sur les différentes possibilités.

5 Bilan personnel

Je tiens à vous remercier pour ces exercices que j'ai trouvé complets. Ils m'ont permis d'avoir un regard plus ouvert sur le monde de la recherche. J'ignorais qu'il existait des bibliothèques en ligne comme OpenAlex ou Scopus si bien organisées permettant d'obtenir des données pertinentes à l'aide de simples requêtes sur des api spécifiques et puissantes. Ce test pratique m'a permis aussi d'utiliser des bibliothèques Python efficaces dont j'ignorais l'existence et de monter en compétences en ce qui concerne la programmation orientée objet. Si j'ai pu tant apprendre en si peu de temps, alors je pense que passer plusieurs semaines au sein de l'ÉTS sera une source d'apprentissage riche qui aura un impact significatif sur mes compétences et sur mon cursus professionnel.