



Projet en Programmation Python

Clément PERENON
Léa REGAZZETTI

Master 1 Informatique
Promotion 2020/2021

Table des matières

Introduction.....	3
I. Spécifications	3
II. Analyse	3
A. Environnement de travail	3
B. Diagramme des classes	4
III. Conception	5
IV. Validation.....	6
A. Tests unitaires : test des méthodes individuellement.....	6
B. Test globaux	9
V. Maintenance.....	12
Conclusion	12
Annexe : Documentation utilisée pour la réalisation du projet	13

Introduction

Ce projet prend place dans la matière programmation avancée en Python du Master 1 d'Informatique de Lyon 2.

Dans ce dossier, nous commencerons par rappeler la problématique du projet choisi ainsi que ces spécifications. Ensuite, nous détaillerons les spécifications du projet et leurs réalisations. Également, nous aborderons la validation de notre programme. Pour finir, nous évoquerons sa maintenance. Vous trouverez le code source de notre logiciel à l'adresse suivante : https://github.com/LeaRegazzetti/Projet_Python_M1_Info

I. Spécifications

Nous avons, parmi les deux sujets, choisi le deuxième, à savoir l'extraction de collocations. Le fonctionnement de notre programme est simple. Au lancement, une interface graphique va permettre à l'utilisateur de taper un mot de son choix, qui sera le « thème » ou le « domaine d'étude » de sa recherche. Par la suite le programme va créer un corpus de document autour de cette thématique. L'utilisateur choisit ensuite le document de son choix dans le corpus et le programme génère un graphe des mots de ce document pour étudier leur co-occurrence.

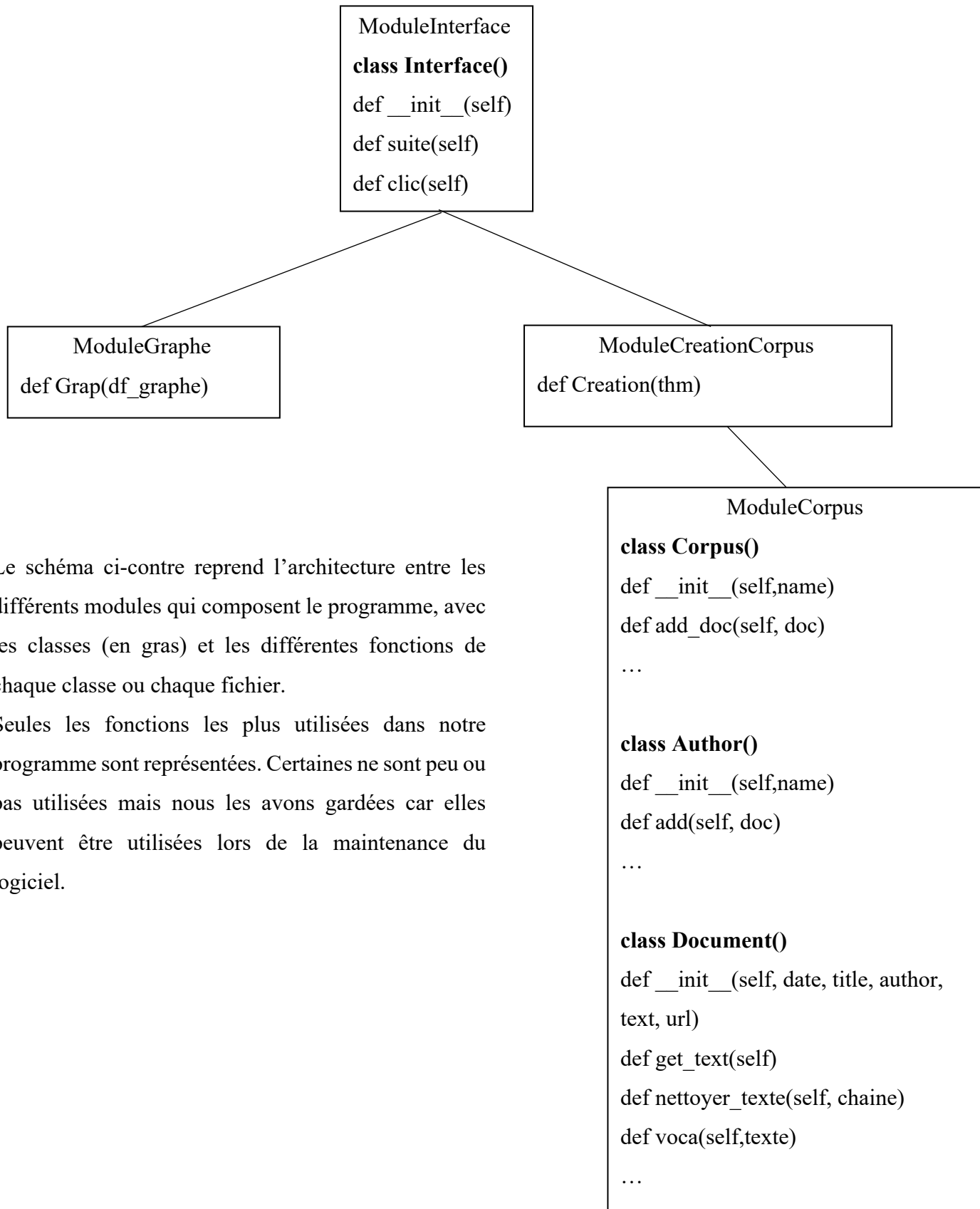
Dans ce projet, nous avons choisi de traiter les résumés de chaque article, et non les articles en eux-mêmes. La volumétrie était suffisamment conséquente pour travailler convenablement. Or, si nous avions à adapter ça, un simple changement de variable lors de la création du corpus nous permettrait de palier ce détail.

II. Analyse

A. *Environnement de travail*

Nous avons développé l'application à l'aide de Anaconda Spyder. Nous sommes habitués à utiliser Spyder sous Anaconda et la facilité de gestion des packages et la performance de cet IDE a été un argument de poids quant au choix de notre environnement.

B. Diagramme des classes



Le schéma ci-contre reprend l'architecture entre les différents modules qui composent le programme, avec les classes (en gras) et les différentes fonctions de chaque classe ou chaque fichier.

Seules les fonctions les plus utilisées dans notre programme sont représentées. Certaines ne sont peu ou pas utilisées mais nous les avons gardées car elles peuvent être utilisées lors de la maintenance du logiciel.

III. Conception

En ce qui concerne la conception du projet, nous nous sommes réparti les différentes spécificités ainsi :

- Les méthodes liées à la création du corpus de documents ont été réalisées conjointement par les deux étudiants.
- Les méthodes liées à la visualisation du graphe ont été réalisées par Clément.
- Les méthodes liées à l'interface ont été réalisées par Léa.

Les algorithmes les plus importants de notre projet, à savoir la mise en forme des données du corpus de document pour l'exécution des graphes ainsi que la création du graphe lui-même, ont été fortement commentés, ainsi, nous avons choisi de ne pas les détailler ici pour éviter la redondance.

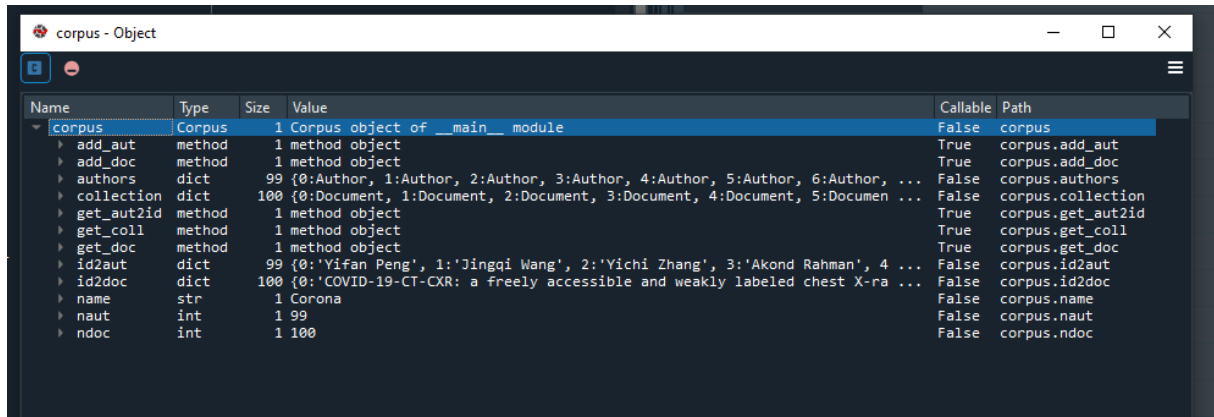
Lors de la conception de notre logiciel, nous avons rencontré quelques problèmes dont notamment l'utilisation de l'identifiant correspondant au titre du document sélectionné par l'utilisateur. En effet, une fonction propre à la librairie Tkinter utilisée pour l'interface permet de récupérer l'identifiant de l'élément sélectionné dans la liste déroulante, toutefois, nous devons utiliser cet identifiant pour afficher le graphe correspondant au document choisi. Or, le contenu de cette variable n'était pas accessible en dehors de la classe. Nous avons donc fait appel à la fonction pour l'affichage du graphe, directement dans la classe `Interface()`.

Un autre problème rencontré a été lors de la création des données d'entrée du graphe. Nous avons initialement calculé pour chaque mot le nombre de fois où il apparaissait dans le document mais cette technique ne nous permettait pas d'obtenir le lien entre chaque mot. Pour résoudre cela, nous avons créé un dataframe de 2 colonnes contenant les mots du document. Dans la première colonne se trouve tous les mots du document excepté le dernier et dans la deuxième colonne se trouve tous les mots excepté le premier. Ainsi, le premier mot est lié avec le deuxième, le deuxième avec le troisième, etc.

IV. Validation

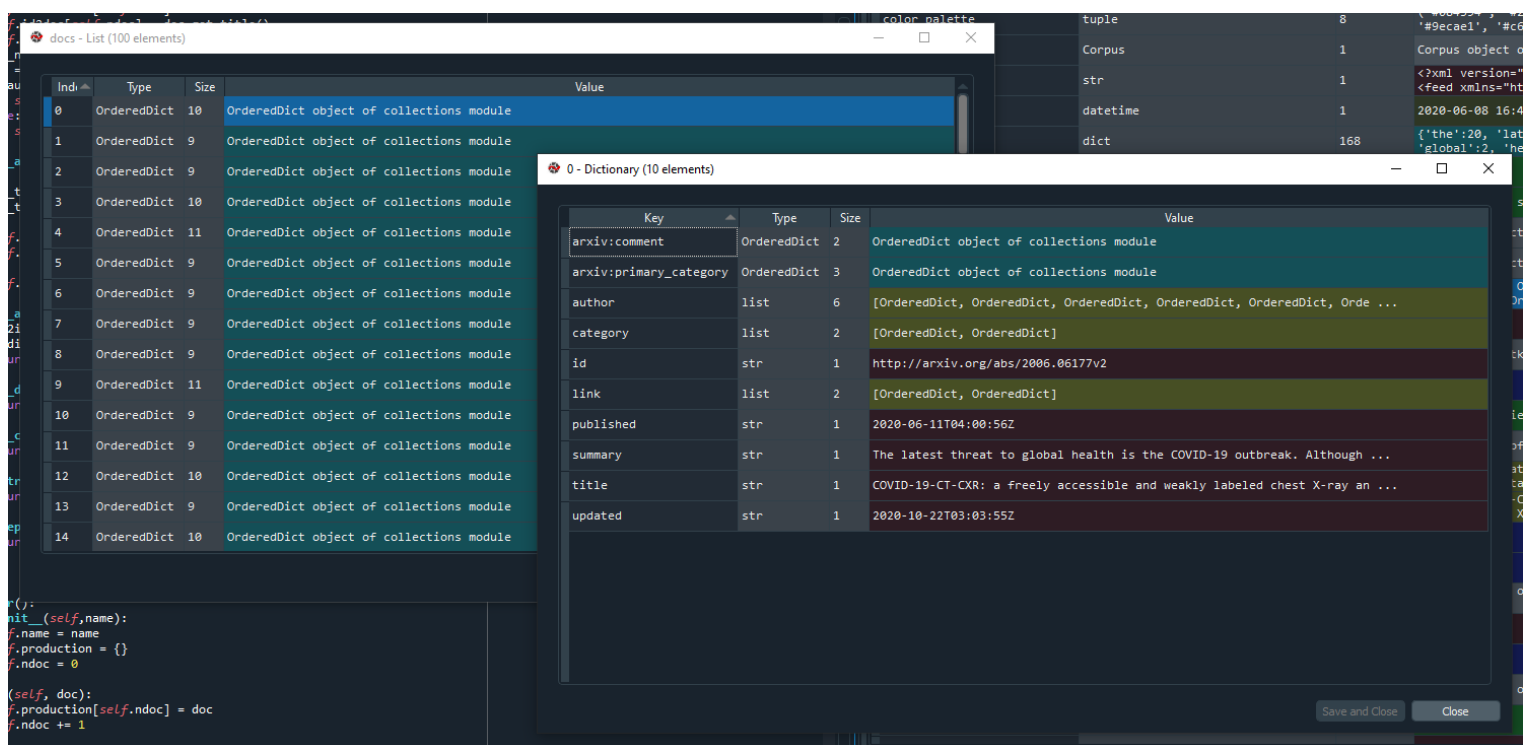
A. Tests unitaires : test des méthodes individuellement

Dans cette partie, nous allons travailler avec les documents issus du thème « Covid19 ». Après création du corpus nous obtenons un objet de cette forme :



Name	Type	Size	Value	Callable	Path
corpus	Corpus	1	Corpus object of _main__ module	False	corpus
add_aut	method	1	method object	True	corpus.add_aut
add_doc	method	1	method object	True	corpus.add_doc
authors	dict	99	{0:Author, 1:Author, 2:Author, 3:Author, 4:Author, 5:Author, 6:Author, ...}	False	corpus.authors
collection	dict	100	{0:Document, 1:Document, 2:Document, 3:Document, 4:Document, 5:Document, ...}	False	corpus.collection
get_aut2id	method	1	method object	True	corpus.get_aut2id
get_coll	method	1	method object	True	corpus.get_coll
get_doc	method	1	method object	True	corpus.get_doc
id2aut	dict	99	{0:'Yifan Peng', 1:'Jingqi Wang', 2:'Yichi Zhang', 3:'Akond Rahman', 4: ...}	False	corpus.id2aut
id2doc	dict	100	{0:'COVID-19-CT-CXR: a freely accessible and weakly labeled chest X-ra ...}	False	corpus.id2doc
name	str	1	Corona	False	corpus.name
naut	int	1	99	False	corpus.naut
ndoc	int	1	100	False	corpus.ndoc

Les documents du corpus sont dans « collection ». On extrait ces données dans la variable « docs ». On observe une liste d'OrderedDict qui sont en réalité des dictionnaires. C'est donc une liste de dictionnaire, eux-mêmes rempli par des champs (capture d'écran ci-dessous). Chaque document est lui-même un dictionnaire.



Indi	Type	Size	Value
0	OrderedDict	10	OrderedDict object of collections module
1	OrderedDict	9	OrderedDict object of collections module
2	OrderedDict	9	OrderedDict object of collections module
3	OrderedDict	10	OrderedDict object of collections module
4	OrderedDict	11	OrderedDict object of collections module
5	OrderedDict	9	OrderedDict object of collections module
6	OrderedDict	9	OrderedDict object of collections module
7	OrderedDict	9	OrderedDict object of collections module
8	OrderedDict	9	OrderedDict object of collections module
9	OrderedDict	11	OrderedDict object of collections module
10	OrderedDict	9	OrderedDict object of collections module
11	OrderedDict	9	OrderedDict object of collections module
12	OrderedDict	10	OrderedDict object of collections module
13	OrderedDict	9	OrderedDict object of collections module
14	OrderedDict	10	OrderedDict object of collections module

Key	Type	Size	Value
arxiv:comment	OrderedDict	2	OrderedDict object of collections module
arxiv:primary_category	OrderedDict	3	OrderedDict object of collections module
author	list	6	[OrderedDict, OrderedDict, OrderedDict, OrderedDict, OrderedDict, Orde ...]
category	list	2	[OrderedDict, OrderedDict]
id	str	1	http://arxiv.org/abs/2006.06177v2
link	list	2	[OrderedDict, OrderedDict]
published	str	1	2020-06-11T04:00:56Z
summary	str	1	The latest threat to global health is the COVID-19 outbreak. Although ...
title	str	1	COVID-19-CT-CXR: a freely accessible and weakly labeled chest X-ray an ...
updated	str	1	2020-10-22T03:03:55Z

Le premier document (à la position 0 dans la liste) du corpus est donc composé des éléments suivants :

0 - Dictionary (10 elements)

Key	Type	Size	Value
arxiv:comment	OrderedDict	2	OrderedDict object of collections module
arxiv:primary_category	OrderedDict	3	OrderedDict object of collections module
author	list	6	[OrderedDict, OrderedDict, OrderedDict, OrderedDict, OrderedDict, Orde ...
category	list	2	[OrderedDict, OrderedDict]
id	str	1	http://arxiv.org/abs/2006.06177v2
link	list	2	[OrderedDict, OrderedDict]
published	str	1	2020-06-11T04:00:56Z
summary	str	1	The latest threat to global health is the COVID-19 outbreak. Although ...
title	str	1	COVID-19-CT-CXR: a freely accessible and weakly labeled chest X-ray an ...
updated	str	1	2020-10-22T03:03:55Z

Ensuite, nous devons sélectionner un document. En l'occurrence ici, nous pouvons sélectionner un indice en « dur » car cette partie à juste pour vocation de montrer que la génération de graphe fonctionne.

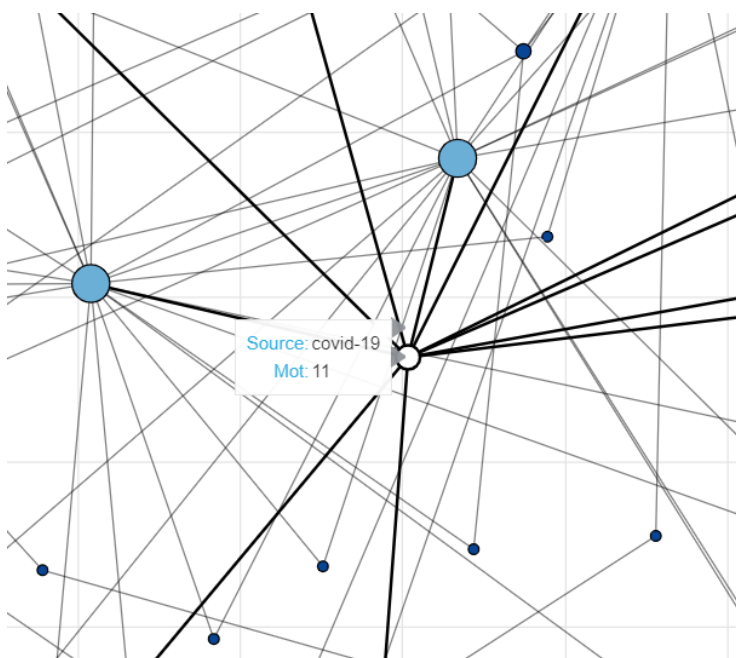
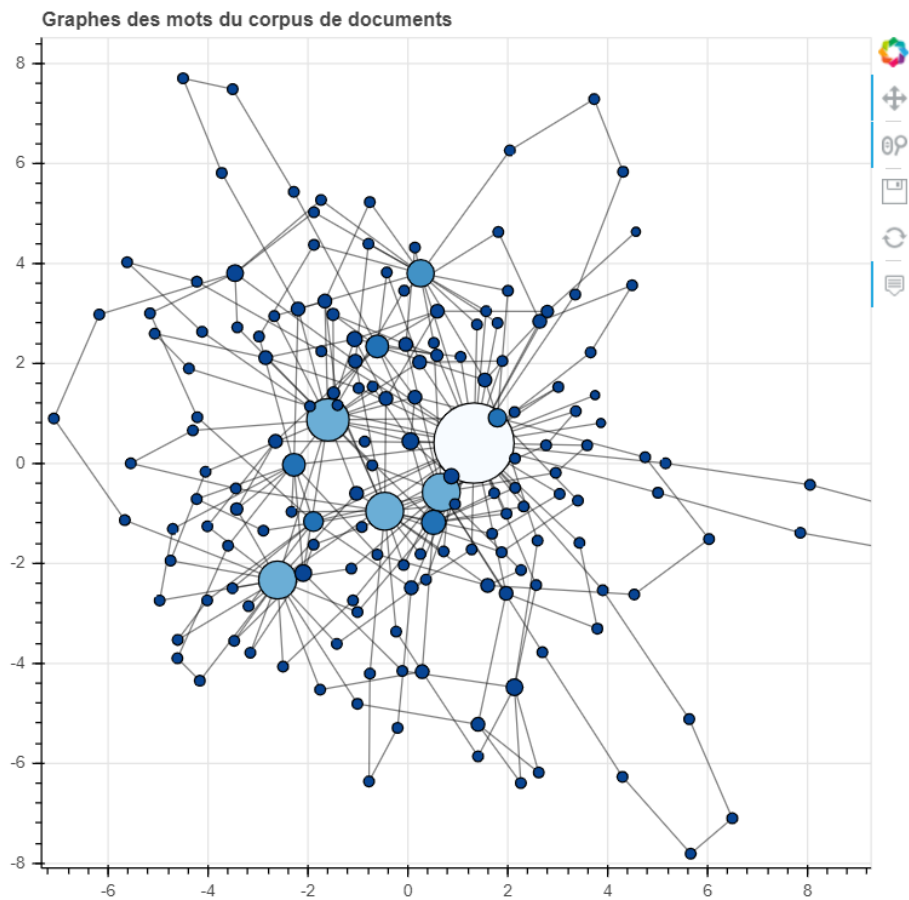
Après transformation de la liste de document en dataframe avec les mots, nous obtenons une liste de dataframes de cette forme :

liste_document - List (100 elements)

Indi	Type	Size	
0	DataFrame	(314, 2)	Column names: source, target
1	DataFrame	(158, 2)	Column names: source, target
2	DataFrame	(267, 2)	Column names: source, target
3	DataFrame	(233, 2)	Column names: source, target
4	DataFrame	(139, 2)	Column names: source, target
5	DataFrame	(175, 2)	Column names: source, target

Dans les dataframes sont stockés les mots, ainsi que ceux avec qui ils sont liés dans le document.

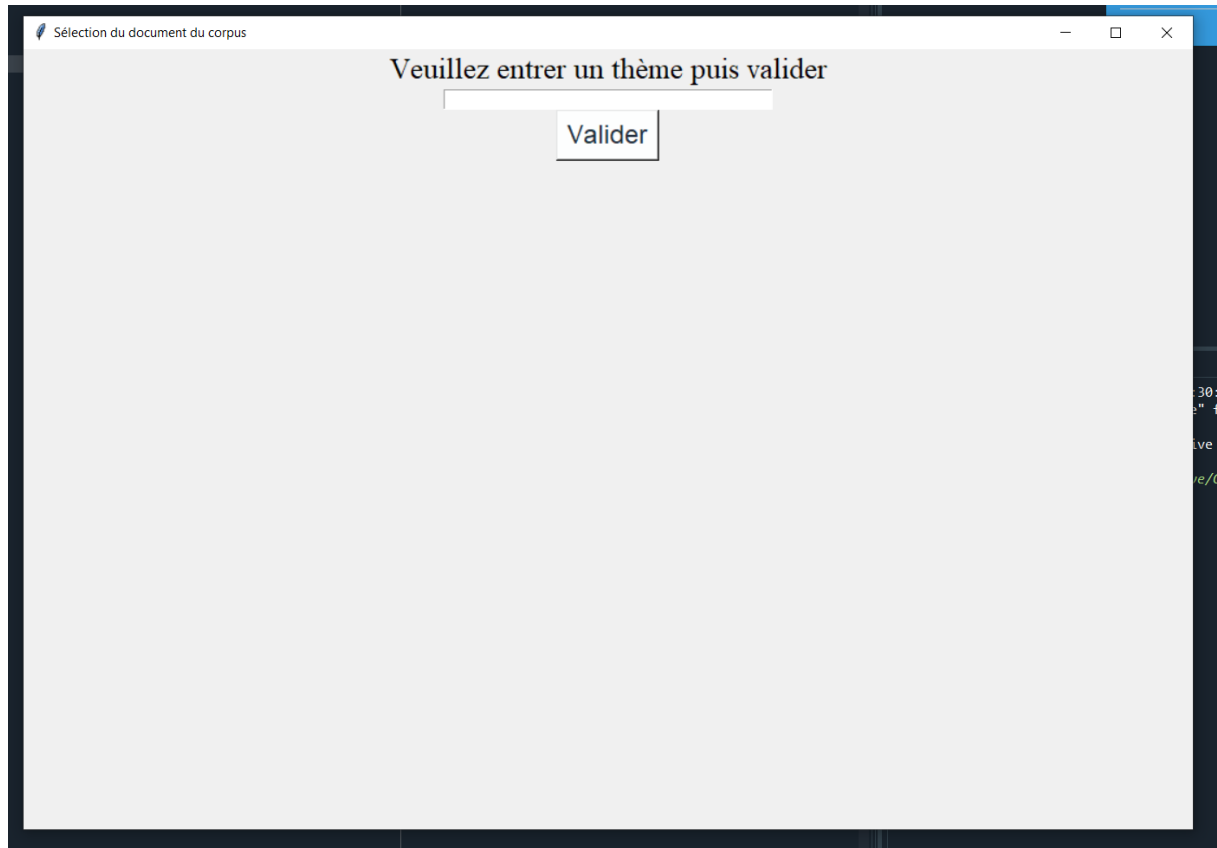
On stocke dans une variable le document sélectionné, et on réalise le graphe avec le dataframe stocké.



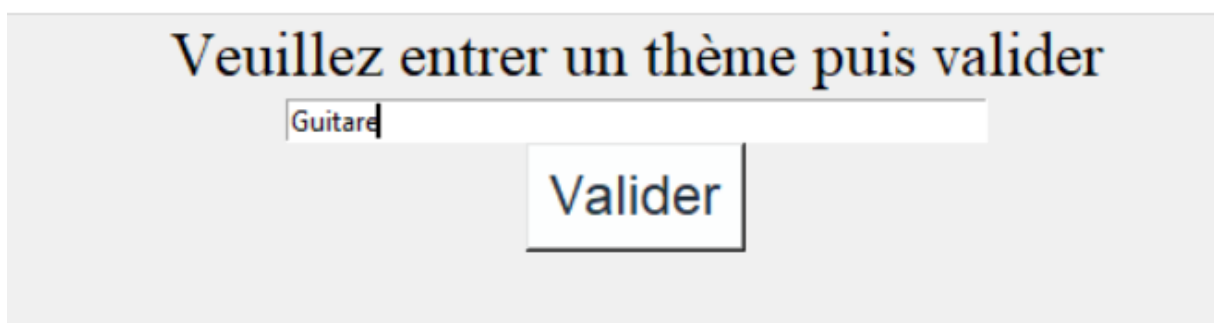
On peut voir que dans le résumé de cet article, le mot covid a été utilisé 11 fois.

B. Test globaux

On exécute le programme. Après lancement, une interface graphique s'affiche nous permettant de saisir le thème que nous souhaitons.



On rentre un thème au hasard :



Le programme récupère les articles en rapport avec le thème, crée le corpus puis nous donne la possibilité de choisir entre tous les articles du corpus dans une liste déroulante.

Veuillez entrer un thème puis valider

Guitare

Valider

Vous avez entré le thème : guitare

Veuillez sélectionner le titre d'un document puis valider

Latent Space Oddity: Exploring Latent Spaces to Design Guitar Timbres

A Digital Guitar Tuner

Automatic Composition of Guitar Tabs by Transformers and Groove Modeling

Guitar Effects Recognition and Parameter Estimation with Convolutional Neural Networks

Simulating a guitar with a conventional sonometer

Clustering of Musical Pieces through Complex Networks: an Assessment over Guitar Solos

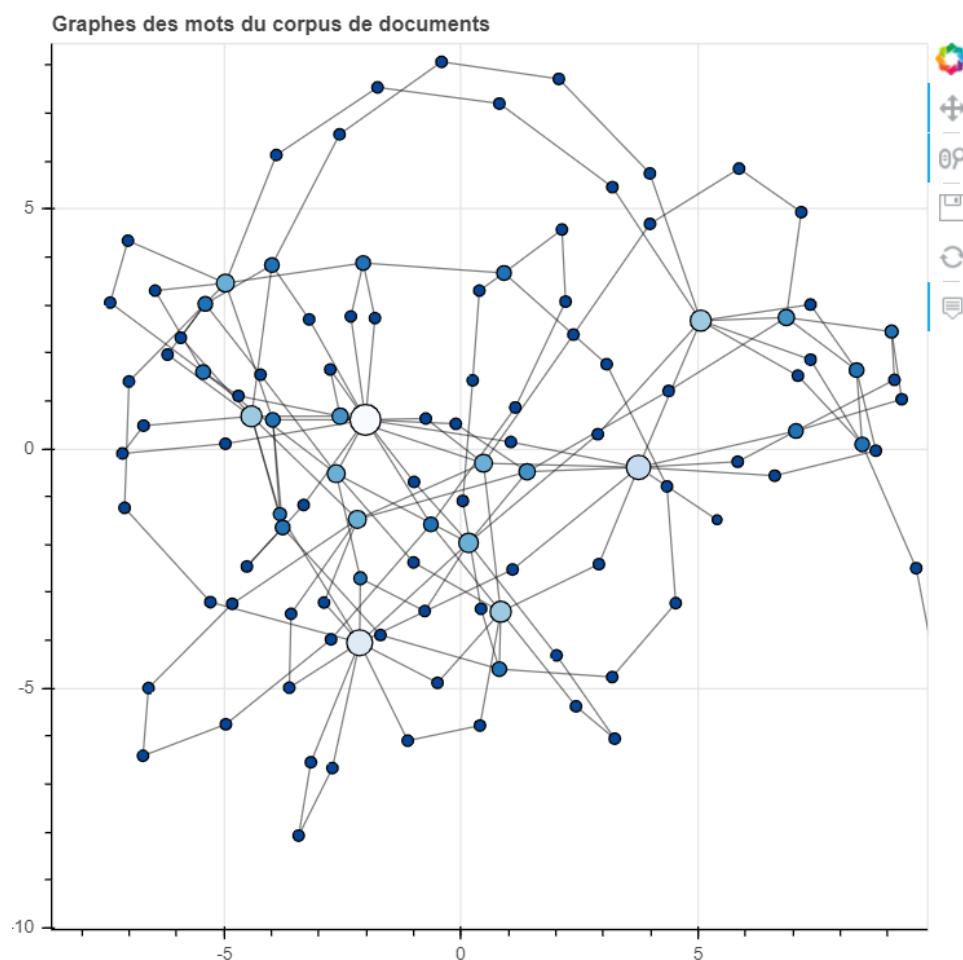
Teaching Fourier Analysis and Wave Physics with the Bass Guitar

Reconstructing the Guitar: Blowing Bubbles with a Pulsar Bow Shock Back Flow

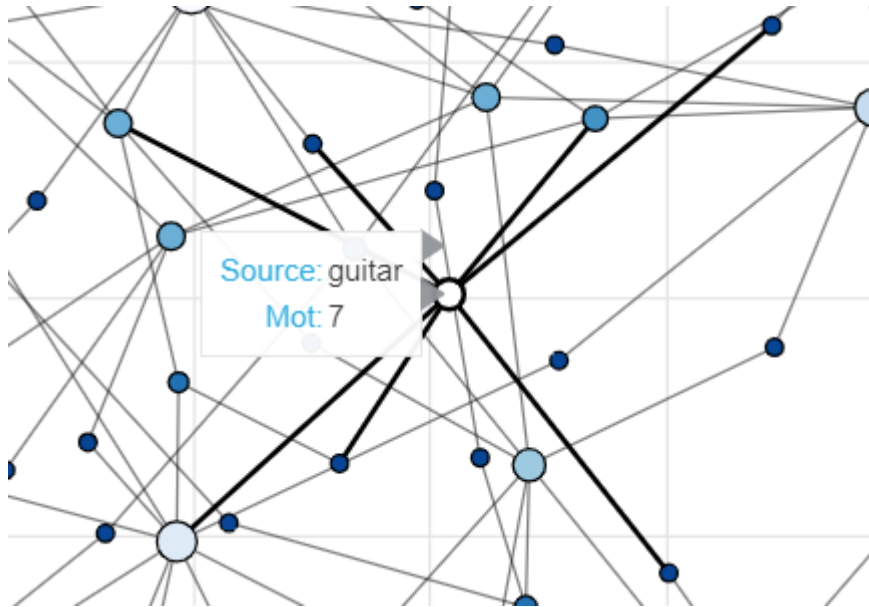
X-ray Emission from the Guitar Nebula

Guitar Solos as Networks

On clique sur valider, et le programme va générer un graphe avec les poids de chaque mot ainsi que les liens entre les mots.



Nous avons la possibilité d'interagir avec le graphe (zoom, de-zoom, enregistrement en image...). Se positionner au-dessus d'un nœud nous donne également le mot en question, le met en surbrillance et épaisse les liens avec les autres nœuds. Il affiche également le nombre de fois qu'il est utilisé dans le document.



V. Maintenance

Concernant les évolutions possibles de notre logiciel, nous pouvons évoquer en premier la gestion du texte rentré par l'utilisateur grâce à l'interface lors de la demande du thème. En effet, à l'heure actuelle, notre logiciel gère uniquement le cas où un seul mot est entré. Il faudrait donc permettre à l'utilisateur d'insérer plusieurs mots ou groupe de mots et adapter cette saisie à la spécificité de la requête permettant d'interroger l'API. Ainsi, il s'agirait de séparer les mots saisis par l'utilisateur et de rajouter l'expression '+AND+all :' entre chaque mot. Voici un exemple de l'url que l'on doit obtenir si l'utilisateur entre les mots 'electron' et 'proton' : http://export.arxiv.org/api/query?search_query=all:electron+AND+all:proton.

Il faudrait aussi pouvoir gérer le cas où l'utilisateur insère un texte saugrenu. Si tel est le cas, un message d'erreur pourrait être affiché grâce à l'interface indiquant que le mot n'existe pas et redemander à l'utilisateur d'insérer du texte.

Nous avons également pensé à l'affichage des mesures de centralité et de communautés liées au graphe affiché. Pour cela, l'utilisation des bibliothèques Python spécialisées dans les graphes serait nécessaire.

Une autre piste à creuser serait l'étude des collocats, à savoir les mots qui coexistent souvent ensemble. Pour cette idée, l'utilisation de la méthode Pointwise Mutual Information, ou PMI, serait à approfondir. De plus, lors de l'affichage du graphe, nous pourrions épaissir les liens entre les collocats afin de mettre en évidence l'utilisation récurrente des termes ensemble.

Conclusion

Pour conclure, nous pouvons dire que ce projet nous a permis de mettre en pratique les connaissances acquises dans le cours et durant nos séances de TP. De plus, à travers un sujet concret, nous avons observé l'enjeu et le potentiel que des graphes peuvent offrir. Le développement d'une application dans son intégralité est également très formateur, nous permettant d'appréhender chaque étape de la méthodologie d'un développeur lors du processus de création d'une application.

Annexe : Documentation utilisée pour la réalisation du projet

- TP réalisé en cours de programmation avancée
- https://docs.bokeh.org/en/latest/docs/user_guide/graph.html
- <https://towardsdatascience.com/data-visualization-with-bokeh-in-python-part-one-getting-started-a11655a467d4>
- <https://pandas.pydata.org/docs/>
- <https://networkx.org/documentation/stable/index.html>
- <https://ivywang.gitbook.io/crash-visulisation/bokeh/5.6-network-graph>
- <https://melaniewalsh.github.io/Intro-Cultural-Analytics/Network-Analysis/Making-Network-Viz-with-Bokeh.html>
- <https://realpython.com/python-data-visualization-bokeh/>
- <https://stackoverflow.com/questions/50402895/retrieving-and-using-a-tkinter-combobox-selection/50405102>
- <https://python.doctor/page-tkinter-interface-graphique-python-tutoriel>
- <https://docs.python.org/3.1/library/tkinter.ttk.html>
- http://www.xavierdupre.fr/app/teachpyx/helpsphinx/c_gui/tkinter.html