Rapport

Projet en Programmation Python

Master 1 Informatique

Enseignants : Julien Velcin et Antoine Gourru

2020 - 2021

UNIVERSITE LYON II

Bochra Barnat et Afaf Ben Haj

SOMMAIRE

INTRODUCTION

Ce projet s'inscrit dans le cadre du cours d'algorithmique et programmation avancée en Python. Il a pour but la réalisation d'un programme Python pour appréhender de manière globale son intérêt et son utilisation, en effet il va permet la validation de la maitrise d'éléments fondamentaux de la programmation (notions de classes et d'héritage, polymorphisme, gestion des collections, modularisation, etc.)

Deux sujets ont été proposés, ici nous aborderons le sujet 1 que nous décrirons un peu plus tard.

Le lien du dépôt Github du projet est le suivant : **https://github.com/afbenhaj/projet-info.** Il contient le code source et l'ensemble des fichiers nécessaires à l'exécution du programme. Vous trouverez également un document requirements.txt permettant de récupérer les librairies nécessaires à l'exécution du programme.

Sujet : analyse comparative de corpus.

Les documents contiennent énormément d'informations pertinentes pour les chercheurs en SHS (sociologues, linguistes, etc.). Néanmoins, la masse de données rend compliquée leur exploration. Pour cela, nous avons créé un outil permettant d'explorer les documents en adoptant une approche comparative.
Dans un premier temps, nous avons proposé différentes manières d'analyser comparativement deux corpus (comparaison des articles issus de Reddit de ceux d'Arxiv). Il s'agit donc ici d'observer l'importance relative d'un ou plusieurs mots dans un corpus vs. un autre corpus. Dans un second temps, nous avons tenté d'observer l'évolution temporelle d'un mot dont l'occurrence est importante donnée. Pour cela, un découpage du champ date était nécessaire. Une interface via Tkinter a donc été créée pour une meilleure visualisation.

1. Les spécifications

Nous allons programmer un outil permettant d'explorer les documents en adoptant une approche comparative.
Cet outil va réaliser l'analyse comparative de deux corpus c'est à dire afficher les mots communs entre les deux textes, l'occurrence et les fréquences des mots dans chaque corpus, avec l'affichage de ces derniers mots choisi. Pour une meilleure pertinence un tri sur les mots trouvés est effectué, en effet les « stop Word », mots et formule courante qui ne sont pas utile pour l'analyse.
Nous observerons également à la demande du client, l'évolution temporelle des mots apparaissant le plus dans les articles issus de Reddit et Arxiv.

2. L'analyse

Ce programme est apparu pour nous diviser en trois parties importantes :

- La première est l'analyse comparative
- La deuxième est l'évolution temporelle des mots.
- Et la troisième la mise en place d'une interface

Nous avons donc fait le choix de se repartir à chacune une partie et de travailler sur l'interface conjointement.
Pour réaliser ces tâches nous avons légèrement travailler sur Jupyter Notebook, essentiellement pour les analyses comparatives et les tests des fonctions de fréquences. C'est un environnement facile et clair pour la programmation avec un affichage direct sous chaque bloque ce qui nous facilite de trouver les erreurs et les corriger.
Mais nous avons essentiellement travaillé sous Spyder, notamment pour bien gérer la bibliothèque Tkinter avec laquelle nous avons travaillé l'interface de notre programme.

Pour commencer la programmation, nous avons choisi de travailler sur le sujet d'actualité le coronavirus.

Notre projet comporte 5 classes :

- Class Corpus : qui a permis d'aller chercher les documents (ndoc).
- Class Author : classe qui donne les informations d'un auteur.
- Class Document : classe mère des deux class RedditDocument et ArxivDocument
- Class RedditDocument : une classe fille de la classe Document permettant de modéliser un document Arxiv
- Class ArxivDocument : une classe fille de la classe Document permettant de modéliser un document Arxiv

3. La conception

Comme déjà mentionner au début. Nous avons divisé le projet en deux grandes pistes.

La première est la partie de comparaison :
Pour cette partie nous avons dans un premier temps traité nos données en effet cette étape est primordiale. Les données textuelles sont difficilement analysables, il a fallu passer par les étapes suivantes :

- Les mettre toute en minuscule pour les comparer correctement.
- Eviter les sauts et retours à la ligne
- Et supprimer, tous les symboles, chiffres et espaces indésirables.
- Enfin il a fallu traiter et supprimer les stopwords une librairie dont l'utilisation est assez simple a été trouvé pour ceci

Ici on souhaite comparer l'apparition des mots dans les différents articles selon leur origine. On à donc crée un Data Frame dont la première colonne correspond aux mots sélectionnés après traitement et la seconde colonne aux nombres de fois qu'il est retrouvé.

Enfin après toutes ces étapes nous pouvons comparer. Pour réaliser cette partie nous avons commencé par comprendre la fonctionnalité des TFIDF (formule permettant de déterminer dans quelles proportions certains mots d'un corpus peuvent être évalués par rapport au reste du texte. Elle utilise le critère de fréquence).
Avant d'arriver au calcule de notre TFIDF, nous devons d'abord calculer le TF puis IDF. Concernant le TF, nous avons créé une fonction mais avons tout, nous avons récupérer nos corpus dans deux paramètres : reddit_txt pour le premier corpus de Reddit et arxiv_txt pour le deuxième corpus d'Arxiv. Ensuite nous avons diviser nos corpus en mot à l'aide de la fonction : split(" ") et nous l'avons stocker dans deux variable : decoup1 et decoup2 . La variable liste reçoit les mots communs entre les deux corpus.

Après avoir réalisé toutes ces étapes, nous pouvons commencer la réalisation de nos fonctions :
def freqTF est la fonction avec laquelle nous allons calculer notre TF (fréquence du mot par rapport au texte). Cette fonction fait appel à deux variables la première est numrepcorpusX (occurrence : le nombre de répétition d'un mot dans un corpus) diviser par le nombre total des mots du corpus decoupCount.

def freqIDF est la fonction avec laquelle nous allons calculer notre IDF (détermine la pertinence d'un texte en considérant un mot-clé précis).

def freqTFIDF est la fonction avec laquelle nous allons calculer notre TFIDF. Elle nous permet d'évaluer l'importance d'un terme dans un corpus. Cette fonction réalise la multiplication des deux valeurs sorties par nos deux fonctions (TF et IDF).

Dans cette partie nous avons également fait des word cloud, graphique qui montre bien l'importance des mots dans chaque corpus.

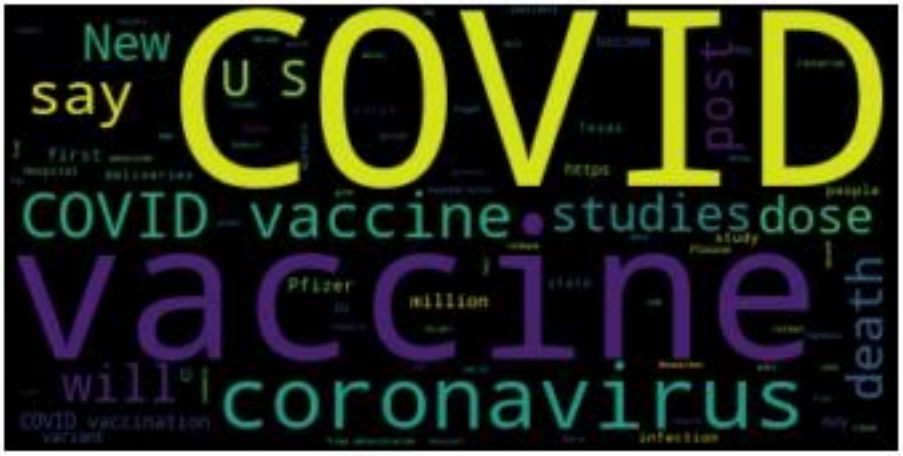

Figure 1 Word cloud Reddit

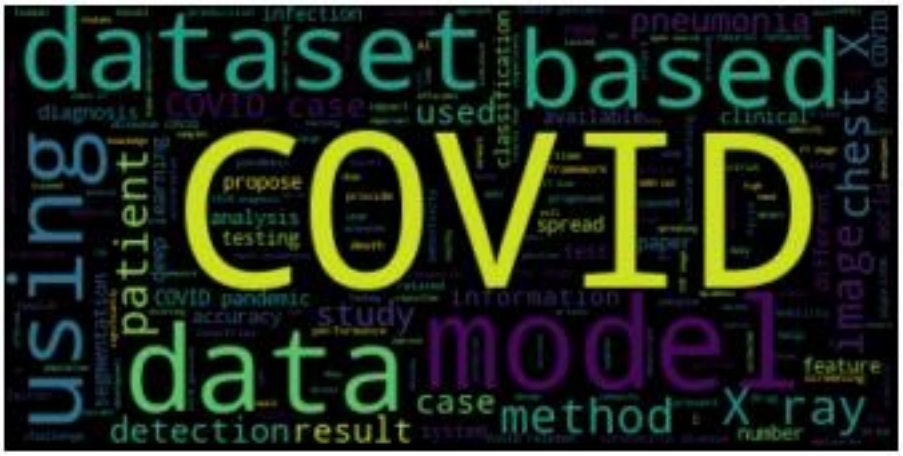

*Figure 2 Word cloud Arxiv**

On voit bien grâce à ces graphiques une large ressemblance des corpus, mais également, des différences. En effet, Arxiv étant une archive ouverte de prépublications

électroniques d'articles scientifiques, on y retrouve des mots tel que model, method, result etc.

- La deuxième est la partie de temporalité des mots :

Celles-ci a mis plus de temps de compréhension, quels mots utilisés pour réaliser une observation intéressante ? Après réflexion nous avons fait le choix de récupérer les 6 premier de chaque dataframe (Reddit, Arxiv) et nous avons créé un dataframe reprenant ces mêmes mots ainsi que les dates où ils apparaissent. Grace à cela, en ne récupérant que les mois et l'année pour une meilleure visibilité nous avons généré un graphique.

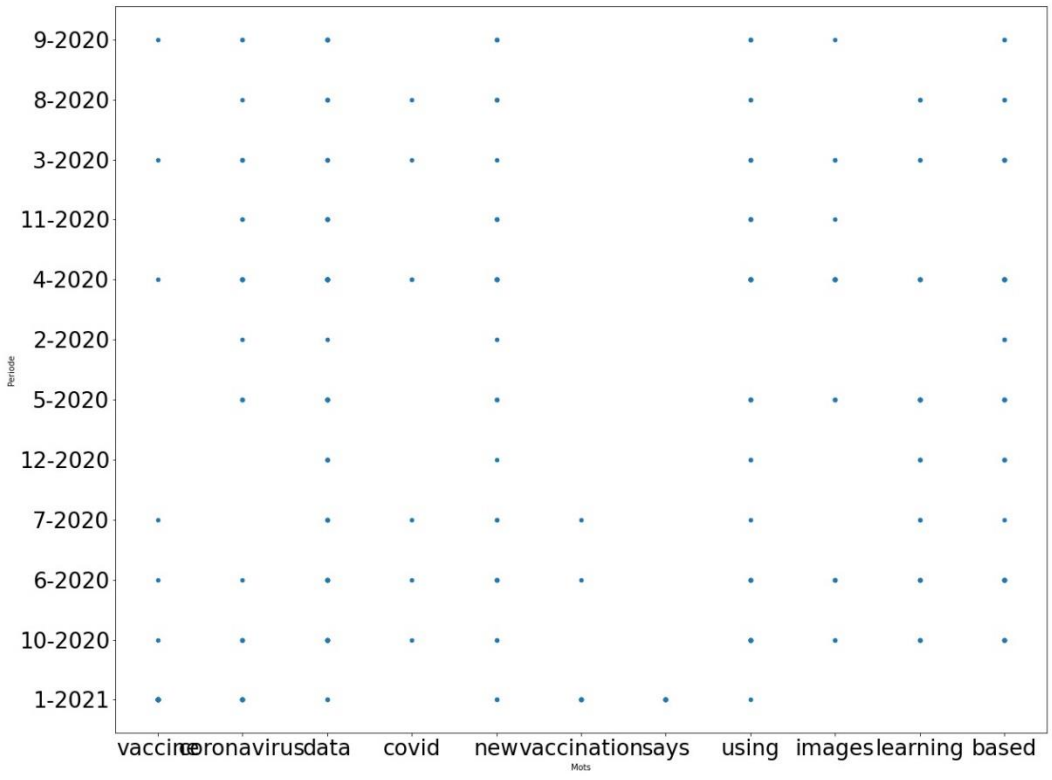

Figure 3 Scatter mots mois et année d'apparition

On voit bien grâce à ce graphique comment les mots se comportent dans l'année si leur utilisation perdure dans le temps ou si au contraire elle est plutôt récente comme pour vaccination par exemple.

Après avoir fini ces pistes, nous sommes passées à l'interface. Notre interface contient 4 boutons et 4 entrées :

- Entree1 : une zone de texte ou nous affichons notre premier corpus Reddit
- Entree2 : une zone de texte ou nous affichons notre deuxième corpus Arxiv
- Entree3 : une zone de texte ou nous allons rentrer un mot que nous souhaitons analyser.
- Sortie1 : une zone de texte ou nous allons afficher nos résultats
- Corpus1 : label
- Corpus2 : label
- Corpus3 : label qui nous demande de taper le mot souhaité dans notre zone de texte entree3
- Comparer : bouton qui réalise l'analyse de nos deux corpus puis l'affiche dans notre zone de texte sortie1.
- Motcommun : bouton qui va réaliser l'affichage des mots commun dans notre zone de texte sortie1.
- Temp : bouton qui va réaliser l'affichage du graphique présentant la temporalité des mots les plus présents.
- Word cloud : bouton qui va réaliser l'affichage des graphiques word cloud réalisé.
- Valider : bouton qui va afficher respectivement les occurrences du mot entrée dans les deux corpus et ses fréquences tfidf dans notre zone de texte sortie1.
- Quitter : bouton pour détruire notre page, c'est à dire fermer notre interface.

Nous n'avons pas rencontré des gros problèmes que nous pouvons les mentionner. Toutes les erreurs eues, étaient des fautes d'inattention, des fautes de frappe ou l'appellation d'une mauvaise variable. Mais en revanche nous aurions en effet souhaités faire plus et pourquoi pas des analyses plus approfondies.

4. La validation

Pour passer d'une étape a une autre, nous validions nos méthodes avant. Nous ne passons pas aux étapes suivantes si notre méthode nous ne donne pas des résultats pertinents. En effet le but du projet étant de rendre un dossier fonctionnel.

Comme nous avons commencé le codage sur Jupyter, nous réalisons le test sur jupyter puis sur spyder.

Voici quelque exemple :

```
Entrée [122]: #creer ma liste des mots sans repetition
              liste = set(decoup1).union(set(decoup2))
              liste

Out[122]: {'',
           '?',
           'allez',
           'bien',
           'bonjouur',
           'et',
           'jespere',
           'mauvais',
           'que',
           'salut',
           'vous'}
```

Image 1

```
Entrée [131]: #calcule freq pour savoir les mots commun (utiliser plusieur fois)
              def freqIDF(documents):
                  import math
                  N = len(documents)

                  idfDict = dict.fromkeys(documents[0].keys(), 0)
                  for document in documents:
                      for word, val in document.items():
                          if val > 0:
                              idfDict[word] += 1

                  for word, val in idfDict.items():
                      idfDict[word] = math.log(N / float(val))
                  return idfDict

Entrée [132]: idf=freqIDF([numrepcorpus1,numrepcorpus2])

Entrée [133]: idf

Out[133]: {'vous': 0.6931471805599453,
           'bien': 0.0,
           '': 0.6931471805599453,
           'que': 0.6931471805599453,
           'bonjouur': 0.6931471805599453,
           '?': 0.6931471805599453,
           'salut': 0.6931471805599453,
           'allez': 0.6931471805599453,
           'jespere': 0.6931471805599453,
           'mauvais': 0.6931471805599453,
           'et': 0.0}
```

Image 2

```
Entrée [138]: #afficher les freqs dans un tableau
              df.head()
```

Out[138]:

	vous	bien		que	bonjouur	?	salut	allez	jespere	mauvais	et
0	0.069315	0.0	0.069315	0.069315	0.069315	0.000000	0.000000	0.069315	0.069315	0.000000	0.0
1	0.000000	0.0	0.000000	0.000000	0.000000	0.138629	0.138629	0.000000	0.000000	0.138629	0.0

Image3

```
Entrée [141]: #calculer la similitude entre deux corpus a la main
              matA=np.array(      list(numrepcorpus1.values())    )
              matB=np.array(      list(numrepcorpus2.values())   )

              x=np.dot(matA,matB.T)
              y=np.sqrt(sum(matA))*np.sqrt(sum(matB))

              x/y
```

Out[141]: 0.5656854249492379

Image 4

Après avoir testé nos méthodes individuellement et avoir les résultats souhaités. Nous passons à l'étape suivante : l'intégration de nos fonctions dans notre interface. Nous l'avons traité point par point.

Au départ, nous avons commencé à afficher les 10 mots communs en cliquant sur le bouton « mots commun » , ensuite l'affichage des analyses en cliquant sur « analyse » puis la temporalité de mot en cliquant sur « temporalité » . Concernant « valider », ce bouton va nous afficher l'analyse d'un mot taper. Finissant par « Quitter » qui va détruire notre page.

Voici un aperçu de notre application :

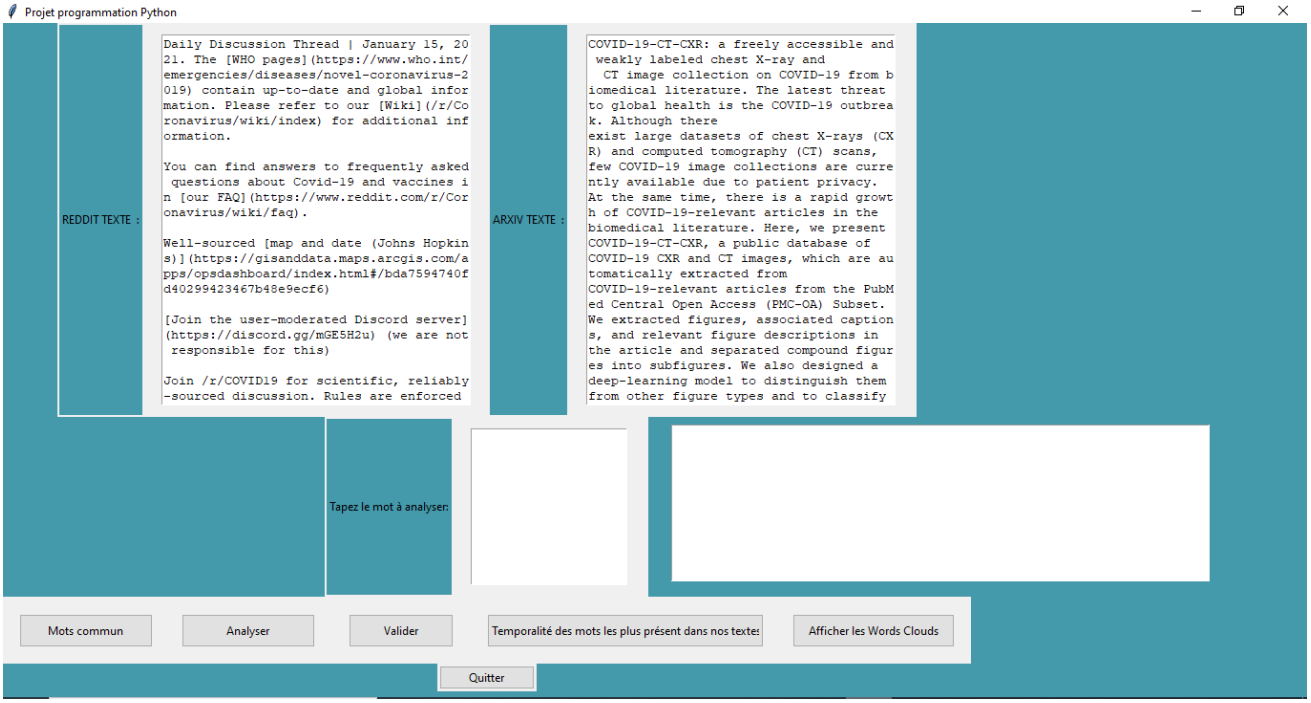

Figure 4 Page interface

Exemple :

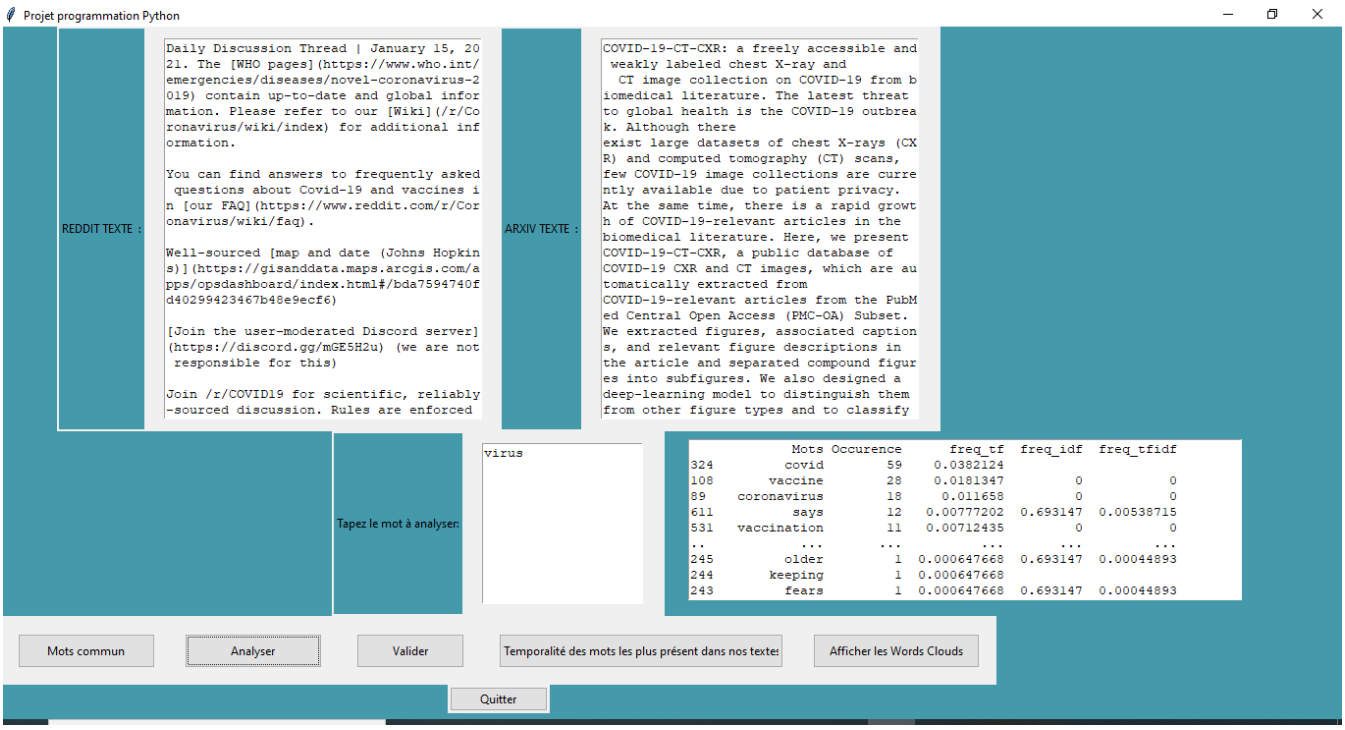

Figure 5Clic sur le bouton "Analyser"

De la même manière pour la seconde partie, nous sommes passées par des graphiques assez aberrants, il a fallu plusieurs tests pour valider celui choisi.

5. La maintenance

Plusieurs évolutions sont possibles sur ce logiciel. Des fonctionnalités pourraient être ajoutées, ajustées comme l'affichage du pourcentage de ressemblance entre les deux textes (déjà réaliser mais pas intégré(image4)), l'affichage d'une phrase, d'un paragraphe qui se répète dans les deux corpus etc.
Il aurait été intéressant de proposer à l'utilisateur d'entrée un mot et d'observer son évolution selon les périodes.
L'interface est aussi améliorable pour avoir une disposition plus claire et une exécution plus efficace, propre visuellement.

CONCLUSION ET PERSPECTIVES

À travers ce projet, nous avons pu consolider nos connaissances acquises et de les enrichir. Nous avons pu comprendre et expérimenter les différentes étapes de la conception d'un programme, en commençant par la spécification jusqu'à maintenance. En plus d'être un projet pédagogique, il a aussi pour nous été ludique et nous a donné beaucoup de liberté dans le code et dans la conception.
Néanmoins, ce projet ne s'est pas terminé de la manière voulue. Nous n'avons pas pu traiter tous les points, mais nous sommes tout de même fières du travail accompli et proposé. Plusieurs de nos fonctionnalités sont intéressantes et d'autres a ajouté le serait d'autant plus comme évoqué dans la partie précédente.