

Analyse de données avec Python

Ce projet est un programme python réalisé dans le cadre d'une demande de l'ANSES (Agence Nationale de Sécurité Sanitaire). C'est au cours d'une conversation familiale informelle qu'une personne de ma famille travaillant à l'ANSES m'a fait part de son besoin. J'ai réalisé qu'avec mes connaissances sur le langage Python je pouvais l'aider. Cette personne a donc formalisé cette demande :

Demande du 24/04/2020 à Sulian :

L'objectif est d'obtenir une liste d'affiliations d'organismes de recherche européens travaillant sur un domaine scientifique particulier. Une base de données bibliographiques sous Endnote a été constituée suite à l'interrogation de base de données bibliographiques externes (Scopus, Pubmed). Un export de ces informations a été fait en fichier txt.

A partir du fichier source Auteurs.txt qui contient donc les références bibliographiques exportées depuis le logiciel bibliographique Endnote, nous souhaitons avoir sous fichier texte .txt, avec les champs séparés par des virgules, et les données dans un même champ séparé par des point virgules et chaque ligne de données séparées par un point d'exclamation :

- La liste des auteurs suivies de leurs affiliations, affiliations regroupées dans un même champ.

Il faut regrouper les mêmes auteurs en un seul champ avec leur occurrence d'indiquée dans un autre champ, occurrence selon le nombre de titres.

- La liste des titres par auteur, titres regroupés dans un même champ.
- La liste des mots-clés par auteur (pour faire corréler ensuite les affiliations avec les mots-clés).

Ce qui donnerait en champs :

Occurrence des auteurs/auteur/ensemble des titres/ensemble des affiliations/ensemble des mots-clés

Mon travail

L'annexe en fin de document présente le programme répondant à la demande que j'ai développé.

Je ne l'ai pas mis en ligne vu que le programme est court.

ANNEXE PROGRAMME PYTHON

```
exemple =
"a1;a12;a13,t1,f11;f12;f13,c11;c12!a1;a22,t2,f2,c21;c22;c23!a3;a22,t3,f31;f32;f3
3,c3!a1,t4,f11,c4!"
liste_cluster = "a22;a1;a3!"

exemple2 = ''

liste_cluster2 = ''

def appartient_oui(l1, l2):
    n = len(l1)
    for i in range(n):
        if appartient(l2, l1[i]): return True
    return False

def liste_ref(chaine):
    l = []
    deb_mot = 0
    n = len(chaine)
    for i in range(n):
        if chaine[i] == "!":
            l.append(chaine[deb_mot : i])
        if chaine[i] == ";":
            l.append(chaine[deb_mot : i])
            deb_mot = i + 1
    return l

def ajouter1 (l1, l2, t):
    h = []
    n = len(l1)
    for i in range(n):
        h.append([l1[i], l2[i], t])
    return h

def ajouter2 (l1, l2, t):
    h = []
    n = len(l1)
    for i in range(n):
        h.append([l1[i], l2[0], t])
    return h

def premier_traitement(chaine, liste_a):
    l = liste_ref(liste_a)
    n = len(chaine)
    a_f_t = []
    notice_pb = ''
    deb_mot = 0
    deb_notice = 0
    na = 0
    nf = 0
    c = 0
    tempa = []
    tempf = []
    for i in range(n):
        if chaine[i] == '!':
            if na != nf:
```

```

        if nf != 1:
            if appartient_oui(tempa, l):
                notice_pb += chaine[deb_notice:i] + '!'
        if nf == 1:
            a_f_t += ajouter2(tempa, tempf, titre)
    if na == nf:
        a_f_t += ajouter1(tempa, tempf, titre)
    deb_notice = i+1
    deb_mot = i+1
    na = 0
    nf = 0
    c = 0
    tempa = []
    tempf = []
if c == 2 and chaine[i] == ',':
    tempf.append(chaine[deb_mot:i])
    nf += 1
    c += 1
if c == 2 and chaine[i] == ';':
    tempf.append(chaine[deb_mot:i])
    nf += 1
    deb_mot = i+1
if c == 1 and chaine[i] == ',':
    titre = chaine[deb_mot : i]
    c += 1
    deb_mot = i+1
if c == 0 and chaine[i] == ',':
    tempa.append(chaine[deb_mot:i])
    na += 1
    c += 1
    deb_mot = i+1
if c == 0 and chaine[i] == ';':
    tempa.append(chaine[deb_mot:i])
    na += 1
    deb_mot = i+1
return (a_f_t, notice_pb)

def appartient(l, m):
    n = len(l)
    for i in range(n):
        if m == l[i]: return True
    return False

def cas_particulier(ligne_mat, liste):
    l_f = ligne_mat[2]
    l_t = ligne_mat[3]
    f = liste[1]
    t = liste[2]
    if not(appartient(l_t, t)):
        ligne_mat[0] += 1
        ligne_mat[3].append(t)
    if not(appartient(l_f, f)):
        ligne_mat[2].append(f)
    return ligne_mat

def ajouter_matrice(mat, liste):
    n = len(mat)
    i = -1

```

```

compt = 0
while (i == -1) and (compt < n):
    if mat[compt][1] == liste[0]:
        i = compt
    compt += 1
if i == -1:
    liste[1] = [liste[1]]
    liste[2] = [liste[2]]
    mat.append([1] + liste)
else: mat[i] = cas_particulier(mat[i], liste)
return mat

def compiler(chaine, liste_a):
    (a_f_t, notices_pb) = premier_traitement(chaine, liste_a)
    n = len(a_f_t)
    matrice = []
    l = liste_ref(liste_a)
    for i in range(n):
        if appartient(l, a_f_t[i][0]):
            matrice = ajouter_matrice(matrice, a_f_t[i])
    return (matrice, notices_pb)

def transformer_liste(liste):
    n = len(liste)
    c = liste[0]
    for i in range(1, n):
        c += ';' + liste[i]
    return c

def appartient1(mat, m):
    n = len(mat)
    for i in range(n):
        if mat[i][1] == m: return True
    return False

def appartient_non(l1, mat):
    n = len(l1)
    for i in range(n):
        if not(appartient1(mat, l1[i])): return True
    return False

def premier_resultat(chaine, liste_a):
    (matrice, notices_pb) = compiler(chaine, liste_a)
    ch = ''
    n = len(matrice)
    for i in range(n):
        ch += str(matrice[i][0]) + ',' + matrice[i][1] + ',' +
transformer_liste(matrice[i][2]) + ',' + transformer_liste(matrice[i][3]) + '!'
    n_b = '!'
    p = len(notices_pb)
    tempa = []
    deb_mot = 0
    deb_notice = 0
    c = 0
    for i in range(p):
        if notices_pb[i] == '!':
            if appartient_non(tempa, matrice):

```

```
        n_b += notices_pb[deb_notice : i + 1]
tempa = []
deb_notice = i + 1
deb_mot = i + 1
c = 0
if c == 0 and notices_pb[i] == ',': 
    c += 1
    tempa.append(notices_pb[deb_mot : i])
if c == 0 and notices_pb[i] == ';':
    tempa.append(notices_pb[deb_mot : i])
    deb_mot = i + 1
return(ch,notices_pb, n_b)

(a,b,c) = premier_resultat(exemple2, liste_cluster2)

data = open("test3_a.txt", "w")
data.write(a)
data.close()
data = open("test3_b.txt", "w")
data.write(b)
data.close()
data = open("test3_c.txt", "w")
data.write(c)
data.close()
```