Vectorization35

February 1, 2019

In [2]:
```
# coding: ANSI

from collections import Counter
from nltk.stem import SnowballStemmer as stm
from nltk.stem import WordNetLemmatizer
import nltk
import time as cl
from sklearn.feature_extraction.text import TfidfVectorizer
import glob
import os as os
from collections import Counter
# nltk.download('all')
# vocab = eval(readtxt("C:\\Users\Thierry\Documents\Telecom ParisTech\PACT
#Obtention de vocab dans la rubrique consacrée au vocabulaire

### Affichage du fichier
```

In [ ]:
```
from nltk.tokenize import WordPunctTokenizer



def tokenize(file):
    word_punct_tokenizer = WordPunctTokenizer()
    with open(file, "r") as file:
            s = file.read()
            wordsList = word_punct_tokenizer.tokenize(s)
            WL=[]
            punctEscape = ',.:;.?,?!()-&éà%$€£<>[]+^¨*ù/â|...1234567890-‘’
            punctEscape = punctEscape + "\"" + "\\"
            for word in wordsList:
                mark = True
                for punct in punctEscape:
                    if punct in word:
                        mark = False
                if mark:
                    WL.append(word)
    return(WL)
```

0.1 Tokenisation du texte

0.1.1 Lemmatization de la liste de string

```
In [ ]: # Normalise les mots en leurs radicaux
        def lemmatization(texte):    # texte est une liste de str
            wordnet_lemmatizer = WordNetLemmatizer()
            Lem=wordnet_lemmatizer.lemmatize
            snowball_stemmer = stm("english")
            Normal=[]


            for mot in texte:
                if snowball_stemmer.stem(mot) != mot:
                    Normal.append(snowball_stemmer.stem(mot))
                elif Lem(mot, pos = 'n') != mot:
                    Normal.append(Lem(mot, pos = 'n'))
                elif Lem(mot, pos = 'v')!=mot:
                    Normal.append(Lem(mot, pos = 'v'))
                elif Lem(mot, pos = 'a') != mot:
                    Normal.append(Lem(mot, pos = 'a'))
                else:
                    Normal.append(mot)
            return Normal       # List de str
```

0.2 Problèmes des Stop Words (SW)

0.2.1 Création d'une liste de SW

```
In [ ]: # fileSW = open("/Users/NAIT/Desktop/stopWords.txt", "r")
        fileSW=open("C:\\Users\Thierry\Documents\Telecom ParisTech\PACT 3.5\Classif
        lines = fileSW.readlines()

        stopWords = []
        for word in lines:
            word = word.rstrip('\n')
            stopWords.append(word)
        #print(stopWords)
```

0.2.2 Élimination des SW

```
In [ ]: # Transforme une liste de str en une liste de str sans les SW
        def StopWordsElimination(list):
            List = [word for word in list if word not in stopWords or word.isdigit
            finalString =""
            for a in List:
                finalString = finalString + " " + a
            return finalString
```

0.3 Définition d'une norme

```
In [ ]: def Dico():
            Vocabstr = readtxt("C:\\Users\Thierry\Documents\Telecom ParisTech\PACT
            Vocab = eval(Vocabstr)
            prepDico = []
            for mot in Vocab:
                prepDico.append([mot,0])
            dico = dict(prepDico)
            return dico,Vocab

        def comptage(texte):

            dictionnaire, Vocab= Dico()

            for mot in texte:
                if mot in dictionnaire:
                    dictionnaire[mot]+=1


            n = len(dictionnaire)
            vecteur = np.zeros(n)

            for i in range(n):

                vecteur[i] = dictionnaire[Vocab[i]]

            return vecteur

        def TFIDF(corpus, vocab):
            vectorizer = TfidfVectorizer(vocabulary=vocab)
            response = vectorizer.fit_transform(corpus)
            CorpusVect = response.toarray()
            return CorpusVect
```

0.3.1 On a besoin d'une liste de mots pour initialiser la TFIDF avec l'attribut vocab. Ce vocabulaire sera récupéré en annexe dans une fonction Vocabulary()

0.4 Conclusion

```
In [ ]: def Vectorize(fichiertxt):          # Vectorisation d'1 fichier texte
            wordsList = tokenize(fichiertxt)
            Normal = lemmatization(wordsList)
            ListeReduite = StopWordsElimination(Normal)
            return ListeReduite #"str"


        def VectorizationOfDataSet():    # La liste de tous les textes
```

```
    corpus = []
    adresseDocAmb = "C:\\Users\Thierry\Documents\Telecom ParisTech\PACT 3.5
    LAmbiances = os.listdir(adresseDocAmb)

    for ambiance in LAmbiances:
        PtrAmbFold = os.listdir("C:\\Users\Thierry\Documents\Telecom ParisT

        for PtrFile in PtrAmbFold:

            AdresseText = "C:\\Users\Thierry\Documents\Telecom ParisTech\PA
            corpus.append(Vectorize(AdresseText))

    return corpus

def VectorisationAmb(): # Vectorisation des textes d'ambiances
    c1 = cl.clock()
    def VectorizationOfDataSetTMP():    # La liste de tous les textes
        corpus = []
        adresseDocAmb = "C:\\Users\Thierry\Documents\Telecom ParisTech\PACT
        LAmbiances = os.listdir(adresseDocAmb)
        Lid = []
        for ambiance in LAmbiances:
            PtrAmbFold = os.listdir("C:\\Users\Thierry\Documents\Telecom Pa
            n = 0
            for PtrFile in PtrAmbFold:

                AdresseText = "C:\\Users\Thierry\Documents\Telecom ParisTec
                corpus.append(Vectorize(AdresseText))
                n+=1
            Lid.append((ambiance,n))
        return corpus,Lid

    corpus,ident = VectorizationOfDataSetTMP()
    reponse = TFIDF(corpus,vocab), ident
    c2 = cl.clock()
    print(c2-c1)    # Petite estimation du temps que ça prend
    return reponse
    # reponse = LVecteurs, ident
    # ident est la liste des couples (Ambiance, nbTextes de l'ambiance)

def Vectorisation(LAdresseText):
    LRed = []
    for AdresseText in LAdresseText:
        LRedi = Vectorize(AdresseText)
        LRed.append(LRedi)
    return TFIDF(LRed,vocab)
```

0.5 Réduction de dimensions : PCA

```
In [ ]: from sklearn.decomposition import PCA
        import numpy as np


        def ReductionDim(X,n):
            c1 = cl.clock()
            pca = PCA(n_components = n)
            pca.fit(X)
            Y = X.copy()
            Transformé = pca.transform(Y)
            c2 = cl.clock()
            print(c2-c1)
            return Transformé,pca
```

0.6 Test de la vectorisation

```
In [ ]: def TestPCA():
            Vec = Vectorisation()
            V = Vec.copy()
            n = 30
            return ReductionDim(V,n)
```

0.7 Annexes

```
In [ ]: ##### Vocabulaire



        def Vocabulary():    # Vectorisation de l'ensemble du corpus

            def EnsembleDesTextes():    # La liste de tous les textes
                corpus = []
                adresseDocAmb = "C:\\Users\Thierry\Documents\Telecom ParisTech\PACT
                LAmbiances = os.listdir(adresseDocAmb)

                for ambiance in LAmbiances:
                    PtrAmbFold = os.listdir("C:\\Users\Thierry\Documents\Telecom Pa

                    for PtrFile in PtrAmbFold:

                        AdresseText = "C:\\Users\Thierry\Documents\Telecom ParisTec
                        vec = Vectorize(AdresseText)
                        corpus.append(vec)

                return corpus
```

```
    def tokenize1txt(brut):

        word_punct_tokenizer = WordPunctTokenizer()
        wordsList = word_punct_tokenizer.tokenize(brut)
        wordsList = [word for word in wordsList if word not in ["'",".",":'
        return(wordsList)



    def EnsembleMots(texte): #type(texte)=list(str)
        Dico=Counter(texte)    # Le dico
        Différents = []
        while Dico != Counter():
            Différents.append(Dico.popitem()[0])
        return sorted(Différents)

    c1 = cl.clock()
    Corp = EnsembleDesTextes()
    brut = " "
    for texte in Corp:
        brut = brut + " " + texte
    SimpleTot = tokenize1txt(brut)
    Vocabulaire = EnsembleMots(SimpleTot)

    c2 = cl.clock()
    print(c2-c1)
    return Vocabulaire

def NvTexte(texte,AdresseCible):
    fichiertxt = open(AdresseCible,mode="x")
    fichiertxt.write(texte)
    fichiertxt.close()

def NvGROSTexte(Tableau,AdresseCible):
    fichiertxt = open(AdresseCible,mode="x")
    fichiertxt.write("[")

    for i in range(len(Tableau)):

        if i != 0:
            fichiertxt.write(",")
        fichiertxt.write("[")

        for j in range(len(Tableau[i])):
            if j != 0:
                fichertxt.write(",")
```

```
            try:
                fichiertxt.write(str(Tableau[i,j]))
            except:
                return i,j

        fichiertxt.write("]")

    fichiertxt.write("]")
    fichiertxt.close()

"""
vocab = Vocabulary()
AdresseCible = "C:\\Users\Thierry\Documents\Telecom ParisTech\PACT 3.5\Voca
NvTexte(str(vocab),AdresseCible)
"""

##########################################################################

##### WebScraping: Constitution de la BDD d'ambiances

"""
Ces fonctions sont notamment adaptées à l'interface de Thierry Bécart.
On récupère les codes Source (codeS) de Descriptionari et on en extrait les
On a effectivement remarqu" que sur ce site, les codes sources sont précéde

A chaque fois qu'un texte est récupéré, il est enregistré comme un fichier

"""

# Traduction d'un fichier en str
def readtxt(adresse):

    with open(adresse, "r") as file:
        s = file.read()
    return s



# Détection d'un texte à lire
def detectStart(codeS,i):
    n = len(codeS)
    motifStart = "data-writing-prompt=\"0\"><p>"
    if n-i > 27:
        Testé = codeS[i:i+27]
        if Testé == motifStart:
            return True
    return False
```

```
# Détection de fin de texte
def detectEnd(codeS,i):
    n = len(codeS)
    motifEnd = "</p>"
    if n-i >4:
        Testé = codeS[i:i+4]
        if Testé == motifEnd:
            return True
    return False

# Ecriture d'un texte
def Ecrit(codeS,i):
    j = 0
    T = ""
    while not(detectEnd(codeS,i+j)):
        T=T+codeS[i+j]
        j+=1
    return T

# Extraction des textes du code Source
def diviseTexte(codeS):
    Corpus = []
    i = 0
    n = len(codeS)
    while i!=n:
        if detectStart(codeS,i):
            i+=27
            Texte = Ecrit(codeS,i)
            Corpus.append(Texte)
            decal=len(Texte) + 4
            i += decal
        i+=1
    return Corpus

# Récupération des textes liés à 1 ambiance donnée
def Textes(Ambiance):
    folder = "C:\\Users\Thierry\Documents\Telecom ParisTech\PACT 3.5\Textes
    adresse = folder + '\\' + Ambiance + "\\" + Ambiance + ".txt"    #Adress
    return diviseTexte(readtxt(adresse))


# Création des fichiers Textes extraits des codes source
def Ecrituretxt(Ambiance,numero,texte):
    folder = "C:\\Users\Thierry\Documents\Telecom ParisTech\PACT 3.5\Textes
    adresse = folder + '\\' + Ambiance + "\\" + Ambiance+str(numero)+".txt"
    fichiertxt = open(adresse,mode="x")
    fichiertxt.write(texte)
    fichiertxt.close()
```

```
    # L'adresse finale est ~\Ambiance\Ambiance(numéro).txt

# Crée tous les fichiers textes d'une ambiance donnée
def AjoutAmbiance(Ambiance):
    c1 = cl.clock()
    corpusAmbiance = Textes(Ambiance)
    for i in range(len(corpusAmbiance)):
        Ecrituretxt(Ambiance,i,corpusAmbiance[i])
    c2 = cl.clock()
    print(c2-c1)

# Renvoie le nombre de fichiers textes d'une ambiance donnée
def NTextAmbiances(LAmbiances):
    LNtextes = []
    for Ambiance in LAmbiances:
        corpusAmbiance = Textes(Ambiance)
        LNtextes.append(len(corpusAmbiance))
    return LNtextes

# Renvoie une liste des listes de textes joints  une même ambiance
def Corpus(LAmbiances):
    corpus = []
    for Ambiance in LAmbiances:
        corpusAmbiance = Textes(Ambiance)
        corpus.append(corpusAmbiance)
    return corpus


AdresseVocab = "C:\\Users\Thierry\Documents\Telecom ParisTech\PACT 3.5\Voca
vocab = eval(readtxt(AdresseVocab))
# AdresseGITThierry ="C:\Users\Thierry\Documents\Telecom ParisTech\PACT 3.5
# AdresseVocabGIT = "pact35\modules\Classification\src\Vocabulaire.txt"


def EcritureGROStxt(adresse, Banque):
    c1 = cl.clock()
    folder = "C:\\Users\Thierry\Documents\Telecom ParisTech\PACT 3.5"
    direction = folder + '\\' + adresse + ".txt"
    fichiertxt = open(direction,mode="x")
    fichiertxt.write("[")

    for i1 in range(len(Banque)):
        if i1 == 0:
            fichiertxt.write("[")
        else:
            fichiertxt.write(",[")

        for i2 in range(len(Banque[i1])):
```

```
            if i2 == 0:
                fichiertxt.write("[")
            else:
                fichiertxt.write(",[")

            for i3 in range(len(Banque[i1][i2])):
                if i3 == 0:
                    fichiertxt.write("[")
                else:
                    fichiertxt.write(",[")

                for i4 in range(len(Banque[i1][i2][i3])):
                    if i4 != 0:
                        fichiertxt.write(",")
                    fichiertxt.write(str(Banque[i1][i2][i3][i4]))
                fichiertxt.write("]")
            fichiertxt.write("]")
        fichiertxt.write("]")
fichiertxt.write("]")
fichiertxt.close()
c2 = cl.clock()
print(c2-c1)
# L'adresse finale est ~\Ambiance\Ambiance(numéro).txt
```