Logiciel de correction orthographique

- TIPE

Vincent Guzniczak

Problématique

- Nous souhaitons, dans notre travail, créer un algorithme de correction uniquement orthographique aussi performant que possible pour un nombre limité de lettres (nous nous limiterons à six lettres).

Points essentiels de l‘algorithme

- Utilisation nécessaire de deux bases de données
- Modification des mot entrés par l‘utilisateur dans un but de correction simplement orthographique probabiliste
- Correction instantanée d‘erreurs classiques avec des boucles correctionnelles

Bases de données

- Ensemble des mots de la langue française (on l‘appellera „Dictionnaire“)
- Banque de mots (notée „Banque“)

Algorithmes préliminaires

- 1/Concernant les bases de données

  Comptage des probabilités d'apparition de chaque mot :

```
def Les_mots(texte):
    return re.findall(r'\w+', texte)

MOTS = Counter(Les_mots(open('Banque').read()))

def Probabilité(mot):
    N = sum(MOTS.values())
    return MOTS[mot] / N
```

2/ Algorithmes utiles pour la suite

(Algorithmes de Tri par insertion de suites imbriquées et de Supression de termes d'une suite (pour ne garder que les résultats intéressants))

```
def Tri(Liste):
    n = len(Liste)
    for i in range(1,n):
        x = T[i][1]
        p = i
        while p > 0 and T[p-1][1] > x :
            T[p] = T[p-1]
            p = p-1
        T[p][1] = x
    return(T)

def Suppression_Finale (L):
     If len(L)>=3 :
          del L[0,len(L)-3]
     return(L)
```

Algorithme principal

```
def correction_un_terme(mot):
    L =list(mot)
    if (mot in Dictionnaire==open("Doctexte")) = True :
        print("Le mot est correct")
    elif mot == 'ca':
        mot = 'ça'
        return(mot)
    else :
        P = list(mot)
        Psup = list(mot)
        Liste = []
        for k in range (0,len(P)):
            P = Psup
            for i in ('a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z') :
                P[k] = i
                L =  "".join(P)
                if (L in dictionnaire=open("Doctexte")) = True :
                    SousListe   = []
                    Probabilité(P) = p
                    SousListe.append(P)
                    Sousliste.append(p)
                    Liste.append(SousListe)
        ListeTriée = Tri(Liste)
        ListeTronquée = Suppression_Finale(Liste)
        ListeFinale = []
        for j in range(0, len(ListeTronquée)):
            ListeFinale.append(ListeTronquée[j][0])
        return(ListeFinale)
```

Exemples de résultats

- L’utilisateur rentre le mot « aimr » au lieu de « aime »
Résultat : [‘aima’, ‘aime’]

-L’utilisateur rentre le mot « aims » au lieu de « aime » :
Résultat : [‘aima’, ‘airs’, ‘aime’]

- L’utilisateur rentre le mot « obsectif » au lieu de « objectif » :
Résultat : [‘objectif’]

- L’utilisateur rentre le mot « paurtant » au lieu de « pourtant » :
Résultat : [‘pourtant’]

-L’utilisateur rentre le mot « vieal » au lieu de « viral » :
Résultats : [‘viral’,’vital’]

Optimisation, Complexité, Sources d‘erreurs

Optimisation : - Correction orthographique conditionnelle
- Précision : double édition de termes

Complexité : Boucle if donnant le nombre de lettres (n) fois 26, à additionner avec les boucles if conditionnelles (p). On obtiendra (26.n+p) en premier lieu.

Sources d‘erreurs : - Probabilité des mots
- Limite de l‘édition simple
- Problèmes dus à la boucle correctionnelle

Algorithme principal à double édition

```
def correction_un_terme(mot):
    L =list(mot)
    if (mot in Dictionnaire==open("regexp")) = True :
        print("Le mot est correct")
    else :
        P = list(mot)
        Psup = list(mot)
        Putile = []
        Liste = []
        for k in range (0,len(P)):
            P = Psup
            for i in ('a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z') :
                P[k] = i
                Putile = P
                for j in range (0, len(P)):
                    P = Putile
                    for f in ('a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z') :
                        P[f] = j
                        L =  "".join(P)
                if (L in fichiertxt=open("regexp")) = True :
                    SousListe   = []
                    Probabilité(P) = p
                    SousListe.append(P)
                    Sousliste.append(p)
                    Liste.append(SousListe)
        ListeTriée = Tri(Liste)
        ListeTronquée = Suppression_Finale(Liste)
        ListeFinale = []
        for j in range(0, len(ListeTronquée)):
            ListeFinale.append(ListeTronquée[j][0])
        return(ListeFinale)
```

Exemples de résultats

- L’utilisateur rentre le mot « aimr » au lieu de « aime »
Résultat : [‘airs’, ’aide’, ‘aime’]

-L’utilisateur rentre le mot « aims » au lieu de « aime » :
Résultat : [‘airs’, ’aide’, ‘aime’]

- L’utilisateur rentre le mot « obsectif » au lieu de « objectif » :
Résultat : [‘objectif’]

- L’utilisateur rentre le mot « paurtant » au lieu de « pourtant » :
Résultat : [‘pourtant’]

-L’utilisateur rentre le mot « vieal » au lieu de « viral » :
Résultats : [‘vital’, ’viens’, ’vient’]

Complexité, limites de l'algorithme

Complexité : rajout de deux boucles for à respectivement n (nombre de lettres du mot) et 26 itérations, soit pour p boucles correctionelles, une complexité de (n^2).26^2)+p en premier lieu.

Limites : -Dispersion des résultats
-Problème de précision de la boucle correctionnelle