Université de Bouira
Département d'Informatique
Module : Traitement Automatique des Langues
Niveau M1-ISIL

Série TP 3 - Suite

Analyse lexicale – Soundex.

Le Soundex

Soundex est un algorithme phonétique d'indexation de noms par leur prononciation en anglais britannique. Le code Soundex consiste pour chaque nom en une association d'une lettre suivie de trois chiffres : la lettre correspond à la 1re du nom, et les chiffres encodent les consonnes restantes. Les consonnes à prononciation similaire ont le même code, donc, par exemple, les lettres B, F, P et V sont toutes codées « 1 ». Les voyelles peuvent influencer le code d'une consonne, mais ne sont jamais codées directement (sauf bien sûr si c'est la première lettre du nom).

L'algorithme exact procède comme suit :

1. Supprimer les éventuelles 'espaces' initiales
2. Mettre le mot en majuscule
3. Garder la première lettre
4. Conserver la première lettre de la chaîne
5. Supprimer toutes les occurrences des lettres : a, e, h, i, o, u, w, y (à moins que ce ne soit la première lettre du nom)
6. Attribuer une valeur numérique aux lettres restantes de la manière suivante :
    - **Version pour l'anglais :**
        - 1 = B, F, P, V
        - 2 = C, G, J, K, Q, S, X, Z
        - 3 = D, T
        - 4 = L
        - 5 = M, N
        - 6 = R
    - **Version pour le français :**
        - 1 = B, P
        - 2 = C, K, Q
        - 3 = D, T
        - 4 = L
        - 5 = M, N
        - 6 = R
        - 7 = G, J
        - 8 = X, Z, S
        - 9 = F, V
7. Si deux lettres (ou plus) avec le même nombre sont adjacentes dans le nom d'origine, ou s'il n'y a qu'un h ou un w entre elles, alors on ne retient que la première de ces lettres.
8. Renvoyer les quatre premiers octets complétés par des zéros.

En effectuant cet algorithme, on obtient avec "Robert" et "Rupert" la même chaîne : "R163", tandis que "Rubin" donne "R150".

L'algorithme soundex est très utilisé en informatique pour **corriger les erreurs orthographiques.**

```
# soundex.py
# Make dictionary numerics to map each letter to its group
groups = ['aehiouwy', #0
    'bfpv', #1
    'cgjkqsxz', #2
    'dt', #3
    'l', #4
    'mn', #5
    'r'] #6

numerics = {'a': '0',  'c': '2', 'b': '1', 'e': '0', 'd': '3', 'g': '2', 'f': '1',
 'i': '0', 'h': '0', 'k': '2', 'j': '2', 'm': '5', 'l': '4', 'o': '0',
 'n': '5', 'q': '2', 'p': '1', 's': '2', 'r': '6', 'u': '0', 't': '3',
 'w': '0', 'v': '1', 'y': '0', 'x': '2', 'z': '2'}

def soundex(name):
    """ soundex module conforming to Knuth's algorithm implementation 2000-12-24 by
      Gregory Jorgensen public domain
    """
    # digits holds the soundex values for the alphabet
    sndx = ''
    firstchar = name[0].upper()
    # name = name[1:] # le reste du mot

    # translate alpha chars in name to soundex digits
    for c in name.lower():
        d = numerics.get(c, '0')
        # duplicate consecutive soundex digits are skipped
        if not sndx or (d != sndx[-1]):
            sndx += d

    # replace first digit with first alpha character
    sndx = firstchar + sndx[1:]

    # remove all 0s from the soundex code
    sndx = sndx.replace('0','')

    # return soundex code padded to len characters
    sndx =  sndx + '0000'
    return sndx[:4]

# test it
words =[ "physique", "physik", "phosphore", "fosfor", "hello"]
for word in words :
    print('Word : ', word)
    code = soundex(word)
    print('Soundex Code: ', code, end='\n\n')
```