UNIVERSITÉ DE LORRAINE | UFR MATHÉMATIQUES ET INFORMATIQUE

M1 NLP

methods4NLP
Exercices

1 Regular Expression

1. Consider the usual alphabet of characters in French. Describe the language represented by each of the following expressions :
   (a) $ab^+|b?a$
   (b) $a\backslash d^*\backslash s^+$
   (c) $\backslash s|[ea]\backslash s$
   (d) $[0-7]^+(,[0-7]^*[1-7])?$
2. Consider the usual alphabet of characters in French texts. Find an R.E. characterizing each of the following languages :
   (a) Numerical values in euros with a maximum of two digits after the decimal point,
   (b) Dates written in a format illustrated by the example 04/05/2004
   (c) Strings with at least twice the word "search" (but not at the beginning or end of the string)
   (d) The flexed forms of the french verb "marcher" (present only)
   (e) Lines ending in a question mark
3. Show that each of the following RE is ambiguous :
   (a) $a((ab)^*cd)^*|a(ababcb^*)^*a^*$
   (b) $aab^*(ab)^*|ab^*|a^*bba^*$
4. In each of the following case, give the result of the search :
   (a) R.E. : "a$\backslash w^*$a" Text : "acddcceeacvvv bbaa"
   (b) R.E. : "[^ a-z]^+ba" Text : "Aaa Abbca bac"
   (c) R.E. : "^ .$\backslash b$" Text : "Je viens.$\backslash n$ Et toi ?"
   (d) R.E. : "$\backslash b$[A-Z].*" Text : "Tu connais Paul et Marie ?"
   (e) R.E. : "[$\backslash s$.] [A-Z]" Text : "Tu connais l'O.N.U. C'est difficile."

2 Exp.Reg. en python

The module re is part of the basic python modules.

```python
import re

z=re.compile(r"[bB][a-zA-Z]*") # chaines commençant par b ou B
                               # suivies de n'importe quelles
                               # lettres de a-z ou A-Z
w=re.compile(r"[\w]+") # mots d'au moins une lettre de \w toutes
                       # ie. toutes les lettres (y compris accents)

phrase = "Zorro zozotte contrairement à Bernardo"

w.findall(phrase) # on cherche tous les w dans phrase
                  # et renvoie une liste de ceux-ci

z.search(phrase) # on cherche la premiere occurence de z
                 # dans phrase

z.match("Bernardo") # renvoie None si aucun prefixe n'est
                    # reconnaissable par l'expression

h=w.finditer(phrase) # comme findall mais renvoie un itérateur
```

The following methods can be applied to a match object :

```python
e = re.compile(r"P[\w]+")
m = e.search("hakuna Patata")

if m:
    m.start() # 7
    m.group() # "Patata"
    m.end()   # 13
    m.span()  # (7,13)
else:
    print("Pas de correspondance")
```

Character groups :

```
.  # tous les caractères sauf \n
\w # les lettres (unicode) qui peuvent être dans un mot,
   # y compris les chiffres et _
\W # les caractères qui ne sont pas dans \w
\s # les caractères d'espacement (espace, tabulation, retour
   # à la ligne...)
\S # les caractères qui ne sont pas dans \s
\d # les chiffres (unicode)
\D # les caractères qui ne sont pas dans \d
\b # caractère vide mais uniquement au début ou à la fin
   # d'un mot
\B # caractère vide mais pas au début et à la fin d'un mot.
\A # début de la chaine de caractères = ^
\Z # fin de la chaine de caractères = $
```

Attention : * , + and ? are greedy, which means that they try to "eat" as many times as possible the previous expression. But, there exists a non greedy version : *? , +? et ??.

More on : `https://docs.python.org/3/library/re.html`

1. Write a python script that takes a regular expression and a text as parameters, then displays all instances of the regular expression. Give an example on the following example : “the recipe calls for 6 strawberries and 2 bananas”

2. It is possible to extract a part of a regular expression. Extend the previous function so that it takes a third argument which is the number of the group to be displayed.
3. Build an access.txt file containing the following text, which corresponds to web consultation data :

```
194.250.170.150 - - [18/Sep/2012:10:23:25 +0200]
\"GET /eq_ped.html HTTP/1.0\" 304 -
193.249.17.4 - - [18/Sep/2012:11:34:23 +0200]
\"GET /igsi/ HTTP/1.1\" 200 11125
193.249.17.4 - - [18/Sep/2012:11:34:27 +0200]
\"GET /images/stSel.gif HTTP/1.1\" 200 1429
```

4. Write a python script that extracts IP addresses and calculates the number of web accesses per minute for each of them
5. Write a python script that asks the user (use input()) for a password and checks its validity according to the following conditions :
   — its size must be at least 8 characters long
   — it must contain at least one number
   — it must contain at least one capital letter
   — it must contain at least one special character : _, !, !, ? or -, but not at the beginning of the chain

   Give comments to the user in case of error.

3 Finite State Automata

1. We consider the transition tables of the following automatons (the usual alphabet in French, the initial state is always 0 and the accepting states are marked with an asterisk). For everyone :
   (a) Give its representation in the form of a graph oriented
   (b) Determine the recognized language

(a)	a	b	ϵ
0	1	2	
1	1	3	
2	$\{2,3\}$		
3^*			0

(b)	a	b	,
0	1	2	
1			3
2	2	2	3
3^*	3	3	

2. We consider the French alphabet. For each language give an automaton that recognizes it :
   (a) Any word ending with aa
   (b) Any word with aaa
   (c) Any word without aaa

(d) The bent shapes of the french verb 'marcher' (present tense)

3. Let consider the regular expression $([\hat{}\ a]^*(a[\hat{}\ b])\,?)^*$

   (a) Build an automaton that recognizes the language defined by this regular expression

   (b) Show that this language is not the same as strings that do not contain "ab"

   (c) Modify the automaton and the regular expression so that the language becomes one

4. Build an automaton that recognizes positive (with +) and negative (with -) decimal digits

5. Show that if L_1 and L_2 are two regular languages, then $L_1 \cap L_2$ is too.

6. We consider the french roots "chat", "chien" and "ami". Build an automaton that recognizes all the deflections (m-f/s-p) of these three words

7. We consider the word "coller".

   (a) Build a automata that recognizes at least the following derivations :
   coller, collable, collabilité, recoller, recollable, recollabilité, décoller, décollable, décollabilité, surcoller, surcollable, surcollabilité, redécoller, redécollable, redécollabilité, surdécoller, surdécollable, surdécollabilité

4 Transducer

1. The **Soundex** algorithm is used in library to encode the name of the authors. Its main advantage It has the advantage of being not very sensitive to variations in the writing of names ; example, Jurafsky, Jarofsky, Jarovsky and Jarovski are all represent by J612.

   Using transducers, perform the following different tasks that make up the algorithm in the order in which they are described (it is assumed that the input text has been divided into words separated by the spacing character) :

   (a) Keep the first letter of the name and delete all non-original occurrences of the letters a, e, h, i, o, u, w, y

   (b) Replace letters other than the first by integers according to the rules :

   $$b, f, p, v \to 1,\ c, g, j, k, q, s, x, z \to 2,\ d, t \to 3,\ l \to 4,\ m, n \to 5,\ r \to 6$$

   (c) Replace the series of identical numbers with a single number, e.g., $666 \to 6$

   (d) Remove the numbers beyond the first 3 or, if there are not enough to go up to 3, complete with zeros

2. Build a transducer to normalize a text (without accents) such as :

   (a) upper-case letters (at the beginning of words) become lower-case

(b) punctuations are kept

(c) in the output, words are separated by ≤ 1 space

3. For each rule, build a transducer to implement it :

   (a) replace “c” by “b” if followed by “x”, i.e., $cx \rightarrow bx$

   (b) replace “a” by “b” if followed by “rs”, i.e., $rsa \rightarrow rsb$

   (c) replace “b” by “a” if followed by “rs” and if followed by “xy”, i.e., $rsbxy \rightarrow rsaxy$

4. Construct a transducer that applies the rule of conjugation of English verbs ending in "c" to the past tense (ex. : panic $\rightarrow$ panicked)