Linguistique de corpus

Outils et méthodes de traitement de corpus

Patrick Paroubek

LIMSI-CNRS
Dépt. CHM - Groupe LIR
Bât. 508 Université Paris XI, 91403 Orsay Cedex
pap@limsi.fr

mercredi 30 novembre 2016 / L3 - Cours 11
Un tagger en bash 2/2

Un outil libre et gratuit our dessiner des cartes conceptuelles, utile pour synthétiser des idées, faire le point sur l'essentiel d'une situation, prendre des notes etc. : freeplane. Pour télécharger le source : `https://sourceforge.net/projects/freeplane/`, et pour un tutoriel sur les cartes conceptuelles `http://blogs.lyceecfadumene.fr/informatique/les-fiches-du-cours/les-fiches-freeplane/`.

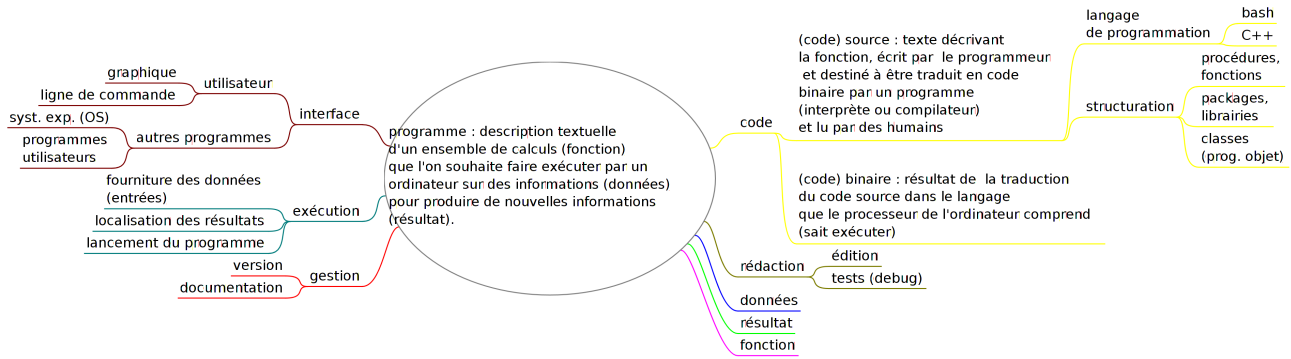

FIGURE – Exemple de carte conceptuelle réalisée avec freeplane.

Reprenons le tagger en bash du dernier cours.

```
#!/bin/bash

# le programme tagger3
# appel: ./tagger3 untexte.txt  dimaju-4.1.1_utf8 untexteavecetiquettesMS3.txt

MONTEXTE=$1
MONDICO=$2
RESULTAT=$3

echo -n "" > $RESULTAT

for w in `cat $MONTEXTE | tr '\012' '\040'`
do
TAG=`egrep -a "^$w " $MONDICO`
if [[ "$TAG" == "" ]];
then
  echo "$w INCONNU" >> $RESULTAT
else
  echo $TAG >> $RESULTAT
done
```

```
tim> head -n 25 montextetagged3_utf8.txt
...etc...
paramétres INCONNU
en PREP PRV:++
entrée, INCONNU
de PREP DTN:pl DTN:sg
quelle DTN:sg PRO:sg ADJ:sg
sortes, INCONNU
que SUB$ SUB REL ADV PRO:sg
représentent-ils INCONNU
>> >> ] ] , , -t PUL -t-on PRV:sg < < - même ADJ:sg « « . . (e) SYM -là ADV &oelig;uvrer
VNCFF &oelig;uvres SBC:pl - ce PRV:sg - ci ADV - t PUL % SYM - t - on PRV:sg ’ ’ [ [
(-) SYM » » - - &oelig;il - de - tigre SBC:sg - t - PUL { { } } -mêmes ADJ:pl (f) SYM ? ?
 -ce PRV:sg &oelig;il-de-tigre SBC:sg ( - ) SYM ( ( (d) SYM (ph. ABR & SYM &oelig;il SBC:sg
: : > > -même ADJ:sg - t - elle PRV:sg ! ! -t-elle PRV:sg ) ) + + (+) SYM (a) SYM / / (b)
SYM (z. ABR ; ; = = &oelig;uvre SBC:sg " " $ SYM ° SYM -t- PUL (c) SYM - t - il PRV:sg ... ..
 - là ADV -t-il PRV:sg - mêmes ADJ:pl -ci ADV 0 CAR 1 CAR 106e ADJ:sg SBC:sg 12e ADJ:sg SBC:s
 1er ADJ:sg 2 CAR 2e ADJ:sg 3 CAR 3e ADJ:sg 4 CAR 4e ADJ:sg 5 CAR 5e ADJ:sg SBC:sg 6 CAR 6e A
 7 CAR 7e ADJ:sg SBC:sg 8 CAR 8e ADJ:sg 9 CAR 9e ADJ:sg a - humaines ADJ:pl a-mi-la ADV a div
ADV a ACJ:sg FGW a - humain ADJ:sg a - humains ADJ:pl a fortiori ADV a giorno ADV a posterior
a priori ADV a. fr. ABR a-humain ADJ:sg a. ABR a - mi - la ADV a. h. all. ABR a maxima ADV a-
ADJ:pl a - humaine ADJ:sg a. b. frq. ABR a. nord. ABR a-humaine ADJ:sg a-humains ADJ:pl a min
aalénien ADJ:sg SBC:sg aalénienne ADJ:sg aaléniennes ADJ:pl aaléniens ADJ:pl SBC:pl ab hoc et
ADV ab intestat ADV ab irato ADV ab ovo ADV aba SBC:sg abaca SBC:sg abacas SBC:pl abacule SBC
abacules SBC:pl abada VCJ:sg abadai VCJ:sg abadaient VCJ:pl abadais VCJ:sg abadait VCJ:sg aba
VNCNT abadas VCJ:sg abadasse VCJ:sg abadassent VCJ:pl abada...etc...
```

Il y a visiblement un problème autour du dernier caractère, le point d'interrogation.

```
tim> cat montexte.txt
Methodologie :
Bien appréhender ce que l'on cherche à faire. Combien de paramétres en entrée,
de quelle sortes, que représentent-ils ?
```

D'ailleurs si l'on regarde la taille du fichier résultat, elle est anormalement grande.

```
tim> ls -l -h montextetagged3_utf8.txt
-rw-r--r-- 1 pap grpiles 8.0M Dec  3 20:47 montextetagged3_utf8.txt
```

Elle est anormalement grande (8 megaoctets !), pour un texte d'origine qui fait 140 octets !

Etrangement, cette taille est aussi la taille du dictionnaire entier.

```
tim> ls -l -h  ls -l -h dimaju-4.1.1_utf8
-rw-r--r-- 1 pap grpiles 8.0M Dec  3 20:47 dimaju-4.1.1_utf8
```

Si l'on regarde la définition des patrons que l'on peut donner comme argument à *egrep*, on apprend que le point d'interrogation est un patron qui filtre 1 seul caractère, mais n'importe lequel ! Si l'on ne veut pas que cela se produise, il faut faire précéder le point d'interrogation par une barre de fraction inverse "\" (*backslash*), qui sera doublée si elle se trouve dans une chaîne èvaluée par l'interprète bash. En effet :

```
tim> w=AAA
tim> echo "^\$w  "
^$w
>tim echo "^\\$w  "
^\AAA
```

Notre tagger devient donc :

```
#!/bin/bash

# le programme tagger4
# appel: ./tagger4 untexte.txt  dimaju-4.1.1_utf8 untexteavecetiquettesMS4.txt

MONTEXTE=$1
MONDICO=$2
RESULTAT=$3

echo -n "" > $RESULTAT

for w in `cat $MONTEXTE | tr '\012' '\040'`
do
    if [[ "$w" == "?" ]];
    then
        PATRON="^\\$w "
        TAG=`egrep -a "$PATRON" $MONDICO`
    else
        TAG=`egrep -a "^$w " $MONDICO`
    fi
    if [[ "$TAG" == "" ]];
    then
        echo "$w INCONNU" >> $RESULTAT
    else
        echo $TAG >> $RESULTAT
    fi
done
```

Le problème est résolu (? est étiqueté ?). Notez qu'il faudra prendre en compte de la même manière tous les autres meta-caractères des patrons de recherche *egrep*, à savoir :
" . , ^ $ \ [ ] * { } "

```
tin> ./tagger4.sh montexte.txt  dimaju-4.1.1_utf8 montextetagged4_utf8.txt
tim> cat montextetagged4_utf8.txt
Methodologie INCONNU
: :
Bien ADV SBC:sg
appréhender VNCFF
ce PRV:sg DTN:sg PRO:sg
que SUB$ SUB REL ADV PRO:sg
l'on INCONNU
cherche VCJ:sg
à PREP
faire. INCONNU
Combien ADV
de PREP DTN:pl DTN:sg
paramétres INCONNU
en PREP PRV:++
entrée, INCONNU
de PREP DTN:pl DTN:sg
quelle DTN:sg PRO:sg ADJ:sg
sortes, INCONNU
que SUB$ SUB REL ADV PRO:sg
représentent-ils INCONNU
? ?
```

Si maintenant nous voulons un tagger "cretin" qui desambiguise systematiquement en prenant la première étiquette, il suffit de sélectionner le second champs avec la commande *cut* − *f*2, nous otenons ainsi le tagger5.

```
#!/bin/bash

# le programme tagger5
# appel: ./tagger5 untexte.txt  dimaju-4.1.1_utf8 untexteavecetiquettesMS5.txt

MONTEXTE=$1
MONDICO=$2
RESULTAT=$3

echo -n "" > $RESULTAT

for w in `cat $MONTEXTE | tr '\012' '\040'`
do
    if [[ "$w" == "?" ]];
    then
        PATRON="^\\$w "
        TAG=`egrep -a "$PATRON" $MONDICO`
    else
        TAG=`egrep -a "^$w " $MONDICO`
    fi
    if [[ "$TAG" == "" ]];
    then
    echo "$w INCONNU" >> $RESULTAT
    else
        ONETAG=`echo "$TAG" | cut -f 2`
        echo "$w $ONETAG" >> $RESULTAT
    fi
done
```

Pour rendre ce tagger un peu plus intelligent, par exemple en préférant choisir Nom comme étiquette si Nom fait partie de la liste, en faisant l'hypothèse que les noms sont plus fréquents ; il suffit d'ajouter un test effectué encore une fois avec *egrep*.

```
#!/bin/bash

# le programme tagger6
# appel: ./tagger6 untexte.txt  dimaju-4.1.1_utf8 untexteavecetiquettesMS6.txt

MONTEXTE=$1
MONDICO=$2
RESULTAT=$3

echo -n "" > $RESULTAT

for w in `cat $MONTEXTE | tr '\012' '\040'`
do
    if [[ "$w" == "?" ]];
    then
        PATRON="^\\$w "
        TAG=`egrep -a "$PATRON" $MONDICO`
    else
        TAG=`egrep -a "^$w " $MONDICO`
    fi
    if [[ "$TAG" == "" ]];
    then
    echo "$w INCONNU" >> $RESULTAT
    else
        NOUNTAG= `echo "$TAG" | tr '\040' '\012' | egrep "SBC:"`
        if [[ "$NOUNTAG" != "" ]];
        then
            echo "$w $NOUNTAG" >> $RESULTAT
        else
           ONETAG=`echo "$TAG" | cut -f 2`
           echo "$w $ONETAG" >> $RESULTAT
        fi
    fi
done
```

Les sorties des trois dernières version de notre étiqueteur.

tagger4	tagger5	tagger6
Methodologie INCONNU	Methodologie INCONNU	Methodologie INCONNU
: :	: :	: :
Bien ADV SBC :sg	Bien ADV	Bien SBC :sg
appréhender VNCFF	appréhender VNCFF	appréhender VNCFF
ce PRV :sg DTN :sg PRO :sg	ce PRV :sg	ce PRV :sg
que SUB$ SUB REL ADV PRO :sg	que SUB$	que SUB$
l'on INCONNU	l'on INCONNU	l'on INCONNU
cherche VCJ :sg	cherche VCJ :sg	cherche VCJ :sg
à PREP	à PREP	à PREP
faire. INCONNU	faire. INCONNU	faire. INCONNU
Combien ADV	Combien ADV	Combien ADV
de PREP DTN :pl DTN :sg	de PREP	de PREP
paramétres INCONNU	paramétres INCONNU	paramétres INCONNU
en PREP PRV :++	en PREP	en PREP
entrée, INCONNU	entrée, INCONNU	entrée, INCONNU
de PREP DTN :pl DTN :sg	de PREP	de PREP
quelle DTN :sg PRO :sg ADJ :sg	quelle DTN :sg	quelle DTN :sg
sortes, INCONNU	sortes, INCONNU	sortes, INCONNU
que SUB$ SUB REL ADV PRO :sg	que SUB$	que SUB$
représentent-ils INCONNU	représentent-ils INCONNU	représentent-ils INCONNU
? ?	? ?	? ?