Syllabation des représentations phonétiques de Brulex et de Lexique

Christophe Pallier*

17 février 1999 (mis à jour le 12 mai 2004)

Cette note décrit l'algorithme utilisé pour syllaber (ou "syllabifier") les représentations phonétiques des bases lexicales du français Brulex et Lexique.

Disons tout de suite qu'il existe plusieurs théories sur la syllabification des groupes de consonnes. Certaines se fondent sur des critères acoustico-phonétiques, d'autres sont d'inspiration plus phonologique. La table 1 (adaptée de Laeufer (1992) dans le chapitre 6 de ma thèse (Pallier, 1994)) présente six propositions.

Pratiquement toutes les théories s'accordent à syllabifier les groupes occlusive–occlusive (OO) entre les deux consonnes : /capture/ est syllabifié en /cap-ture/. Par contre, les théories sont en désaccord sur le statut des groupes occlusive-fricative (OF) : /capsule/ est-il /ca-psule/ ou /cap-sule/ ?

La syllabation que nous proposons pour Brulex et Lexique n'est donc qu'un choix parmi plusieurs possibles (essentiellement fondé sur nos intuitions personnelles !). Elle corresponds à celle de Pulgram sur les exemples de la table 1. Notre idée n'a jamais été d'imposer, avec Lexique, une syllabation. Cette note décrit l'algorithme afin permettre à ceux qui le désirent de corriger selon leurs besoins la syllabation que nous avons proposé.

Si vous utilisez cet algorithme, ou le modifiez, ou détectez des problèmes, je vous serais reconnaissant de me le faire savoir.[1]

*http://www.pallier.org. Voir aussi http://www.lexique.org. Tout commentaire est bien venu.

[1]Mon email est disponible sur le site www.pallier.org

Table 1: Différentes propositions de syllabifications de mots français (d'après Laeufer, 1992)

		Grammont	Delattre		Pulgram Malmberg	Noske	Levin
			apt.	force			
OL	*caprice*	-pr	-pr	-pr	-pr	-pr	-pr
	atlas	-tl	-tl	-tl	t-l	t-l	t-l
ON	*technique*	-kn	-kn	-kn	k-n	k-n	k-n
OF	*adverbe*	-dv	-dv	-dv	d-v	d-v	d-v
OO	*structure*	-kt/k-t	k-t	k-t	k-t	k-t	k-t
FL	*casserole*	-sr	-sr	-sr	s-r	s-r	-sr
	disloque	-sl	-sl	s-l	s-l	s-l	s-l
	influent	-fl	-fl	-fl	-fl	f-l	-fl
FN	*transmis*	-sm	s-m	s-m	s-m	s-m	s-m
FF	*blasphème*	-sf/s-f	s-f	s-f	s-f	s-f	s-f
FO	*diphtongue*	f-t	f-t	f-t	f-t	f-t	f-t
NL	*minerai*	-nr	-nr	-nr	n-r	n-r	n-r
NN	*calomnie*	-mn/m-n	-mn	-mn	m-n	m-n	m-n
NF	*hameçon*	m-s	-ms	-ms	m-s	m-s	m-s
NO	*samedi*	m-d	m-d	-md	m-d	m-d	m-d
LL	*galerie*	-lr/l-r	-lr	-lr	l-r	l-r	l-r
	berlue	-rl/r-l	r-l	r-l	r-l	r-l	r-l
LN	*calmant*	l-m	l-m	-lm	l-m	l-m	l-m
LF	*répulsif*	l-s	l-s	-ls	l-s	l-s	l-s
LO	*culbute*	l-b	l-b	-lb	l-b	l-b	l-b

O = occlusives ; F = fricatives ; N = nasales ; L = liquides

Table 2: Règles de syllabation simplifiées

Chaine	⟶	Syllabation	Exemple(s)
VV	⟶	V–V	poete [po-et]
VCV, VLV, VYV	⟶	V–CV	cadeau [ca-do], calot [ca-lo], aboyer [a-büa-ïé]
VYYV	⟶	VY–YV	bouilloire [buï-üaR]
VCYV, VLYV	⟶	V–CYV	tatouer [ta-tüé], polluer [po-lÿé]
V [td]RV V	⟶	V–CCV	autrement [o-tre-mâ]
VCLV (où C ∈ [pkbgfs/vzj])	⟶	V–CLV	caprice [ka-pRis]
VCCV, VLCV, VLLV, VYCV, VYLV	⟶	VC–CV	capture [kap-ture], costume [kos-tum], galerie [gal-Ri], vieillerie, [vï_ï-Ri], atlas [at-las], madeleine [mad-l_n]
VXXXV	⟶	VX–XXV	astrophysique [as-tRo...]
VXXXXV	⟶	VX–XXXV	obstruer [op-stRy-é], octroyer [ok-tRüa-ïé]

Notations:
V = voyelles (incluant schwa), *sans* les semi-voyelles
C = toutes les consonnes sauf liquides et semi-voyelles
L = liquides {l, R}
Y = semi-voyelles {ü, ï, ÿ}
X = C ou L ou Y

La table 2 résume les principales règles de syllabation que nous nous proposons d'appliquer. Plusieurs remarques méritent d'être soulignées :

- On est obligé de distinguer les semi-voyelles des voyelles: tuer [tüé] vs clouer [klu-é]
- On est obligé de distinguer les semi-voyelles des liquides: polluer [po-lüé] vs. galerie [gal-Ri]
- On a supprimé les schwa finaux des mots multisyllabiques: notre (notRˆ) → [notR] arbre (aRbRˆ) → [arbR]
- Certains choix de syllabation sont certainement discutables, tels que: stagner → [stag-né], castor → [kas-toR], astro → [as-tRo],

Le script de syllabation fournit en annexe explicite toutes les règles.

Nous n'avons pas systématiquement priviligié la règle de l'attaque maximal ou le principe de sonorité.

Nous avons décidé qu'une syllabe non finale contenait au plus une consonne en coda. Par exemple, 'exploit' devient 'ek-sploit'. Ceci est conforme avec la règle de l'attaque maximale même si cela viole le principe de sonorité.

Il nous a aussi fallu prendre une décision vis-à-vis des schwa finaux. Nous avons décidé de les supprimer des représentations phonétiques. Ainsi 'arbre' est pour nous un monosyllabe.

Finalement, notez que les représentations phonétiques de Brulex et de Lexique ne sont pas strictement identiques (cf. par exemple 'fluide' qui contient une voyelle 'u' selon le premier, et un semivoyelle selon le second). C'est pourquoi les syllabations ne sont pas toujours en accord...

Réalisation pratique

L'algorithme est réalisé par un script `syllabation.awk` écrit dans le language Awk. Le programme libre gawk (www.gnu.org/software/gawk/gawk.html) permet de le faire fonctionner.

L'intérêt de Awk est que les expressions régulières permettent d'exprimer les règles de façon très lisible.

Les représentations phonétiques de Brulex et Lexique utilisent des codages différents. Par défaut, le script suppose que le codage de Lexique (en fait de LAIPTTS) est utilisé ; en mettant 'brulex' dans la variable 'code', le cadage de Brulex est employé.

Ce script peut être utilisé interactivement, en tapant simplement :

```
gawk -f syllabation.awk
```

puis en entrant des représentations phonétiques (code Lexique).

Pour syllaber le fichier brulex.txt:

```
gawk -vphons=2 -vcode=brulex -f syllabation.awk brulex.txt
```

et pour syllaber lexique260_graph.txt:

```
gawk -vphons=2 -f syllabation.awk lexique260_graph.txt
```

Si l'utilitaire '`make`' est disponible, il suffit de taper 'make' pour syllaber les deux fichiers.

Le commande '`make test`' fournit les syllabations des mots listés dans le fichier `mots_test.txt`. Le résultat est dans le fichier `tests.txt`.

Les fichiers qui accompagnent `syllabation.awk` sont :

- syllabation.pdf: description (sommaire) des regles de syllabation

- brulex.txt et lexique260_graph.txt : fichiers contenant les formes orthographique et phonetiques de brulex et la table graphemes de lexique.
- Makefile: règles pour l'utilitaire 'make'.
- mots_test.txt : liste de mots dont on veut verifier les syllabations
- tests.txt : resultats de la syllabation (selon brulex, puis selon lexique)

Code source du script `syllabation.awk`

Voici le script awk qui reéalise cette syllabification.

```
#!/usr/bin/gawk -f
#
# This script reads a tab-separated file and syllabifies the columns pointed to b
#
#
# Author: Christophe Pallier (christophe.pallier@m4x.org)
#
# License: GNU (cf. http://www.gnu.org)
#
# Last update: 13 May 2004
# (original date: the first version of this script was written during
#  my dissertation, in 1993)
#
# 2004/05/13: merge of the Brulex & Lexique versions
#             correction for the 'j'->'Z' (viellerie) in Lexique
#             add rule '[td]R' (comment on 'autrefois' by Sprenger-Charolles)
#
#
#
# Note: changed \377 into y-umlaut to run under DOS (bug gawk).

BEGIN {
  FS="\t";
  OFS="\t";

  if (code=="brulex") {
    V="[aiouyîâêôû^eEéAO_]"; # vowels
    C="[ptkbdgfs/vzjmnN£]"; # consonants except liquids & semivowels
    C1="[pkbgfs/vzj]";
    L="[lR]"; # liquids
    Y="[ïü\377]"; # semi-vowels \377 stands for y-umlaut
    X="[ptkbdgfs/vzjmnN£xlRïü\377]"; # all consonants
  } else { # code == LAIPTTS)
    V="[iYeE2591a@oO§uy*]";   # Vowels
    C="[pbmfvtdnNkgszxSZGh]";  # Consonants except liquids & semivowels
    C1="[pkbgfsSvzZ]";
```

```
    L="[lR]"; # liquids
    Y="[j8w]"; # semi-vowels
    X="[pbmfvtdnNkgszSZGlRrhxGj8w]";   # all consonants, including semivowels
   }
   if (phons==0) phons=1;
}

{
 a=$phons;
 n=1
}

{
   while (i= match (a, V V)) {
    a=substr(a,1,i) "-" substr(a,i+1,length(a)); n++; }

  while (i= match(a, V X V)) {
    a=substr(a,1,i) "-" substr(a,i+1,length(a)); n++}

  while (i=match(a, V Y Y V)) {
    a=substr(a,1,i+1) "-" substr(a,i+2, length(a)); n++}

  while (i=match(a, V C Y V)) {
    a=substr(a,1,i) "-" substr(a,i+1, length(a)); n++}

  while (i=match(a, V L Y V)) {
    a=substr(a,1,i) "-" substr(a,i+1, length(a)); n++}

  while (i=match(a, V "[td]R" V)) {
    a=substr(a,1,i) "-" substr(a,i+1, length(a)); n++}

  while (i=match(a, V "[td]R" Y V)) {
    a=substr(a,1,i) "-" substr(a,i+1, length(a)); n++}

  while (i=match(a, V C1 L V)) {
    a=substr(a,1,i) "-" substr (a,i+1,length(a)); n++}

  while (i=match(a, V X X V)) {
    a=substr(a,1,i+1) "-" substr(a,i+2, length(a)); n++}

  while (i= match(a, V X X X V)) {
    a=substr(a,1,i+1) "-" substr(a,i+2,length(a)); n++}

  while (i=match(a, V X X X X V)) {
    a=substr(a,1,i+1) "-" substr(a,i+2,length(a)); n++}

  while (i=match(a, V X X X X X V)) {
    a=substr(a,1,i+1) "-" substr(a,i+2,length(a)); n++}
```

```
86  # suppress the final schwa (^) in some multisyllabic words
87  # notr^ -> notR
88  # ar-bR^   =>  aRbR
89    b=gensub(/-([^-]+)\^$/,"\\1",1,a) ;
90    if (b!=a) { # there is a schwa to delete
91      a=b;
92      $phons=substr($phons,1,length($phons)-1);
93      n-;
94        }
95  # meme chose quand schwa='*'
96    b=gensub(/-([^-]+)\*$/,"\\1",1,a) ;
97    if (b!=a) { # there is a schwa to delete
98      a=b;
99      $phons=substr($phons,1,length($phons)-1);
100     n-;
101       }
102
103
104 # compute the CVY skeleton
105   sk= " ";
106   for (i=1;i<=length(a);i++) {
107     ph=substr(a,i,1);
108     if (ph~V) sk=sk"V";
109     else if ((ph~C)||(ph~L)) sk=sk"C";
110     else if (ph~Y) sk=sk"Y";
111     else sk=sk ph;
112   }
113 }
114
115 { print $0,a,n,sk }
```