Quelques exemples d'utilisation de Lexique avec Awk

Christophe Pallier*

9 Octobre 2001 (dernière mise à jour: 16 mai 2004)

Lexique est une base de données lexicale disponible sur le site www.lexique.org. Elle contient plusieurs tables qui peuvent être interrogées à partir du site. Toutefois, les possibilités offertes par ces recherches en ligne sont limités. Par exemple, il n'est pas possible de faire une recherche restreinte aux résultats de la recherche précédente. De plus, Internet étant ce qu'il est, les requêtes peuvent parfois s'avérer particulièrement lentes. Pour utiliser efficacement Lexique, il est donc conseillé de télécharger sur son ordinateur les fichiers correspondants à chaque table et d'effectuer les recherches avec des outils appropriés.

Ce documentd décrit "Awk", un outil qui permet d'effectuer des recherches, simples ou complexes, dans des fichiers textes tels que les tables de Lexique. Le but n'est pas de vous enseigner Awk (il existe des livres pour cela, et la documentation disponible en ligne sous Linux par 'info gawk' est très bien faite). Plus modestement, nous voulons vous donner un petit aperçu de ce que vous pourrez faire avec cet outil. A vous de juger si vous désirez l'apprendre [1].

Si vous travaillez sous Linux ou MacOS X, vous disposez déjà de gawk, la version gratuite et libre de Awk publiée par le projet GNU : ouvrez un terminal, et vous êtes prêt à travailler. Si vous utilisez Windows, suivez les instructions de cette note ([2]) pour installer gawk et les autres outils nécessaires pour tester les exemples donnés dans ce document.

1. Mon expérience personnelle est la suivante : j'ai découvert Awk il y a plus de dix ans et je l'utilise quasi-journellement. J'ai converti à Awk de nombreux collègues, qui ne s'en plaignent pas. C'est un outil parfaitement adapté aux petites tâches de programmation utiles en psycholginguistique : recherche de matériel expérimentale, randomisation, analyse de données, etc... Je pense que bien des opérations pour lesquelles les gens utilisent Excel ou Access sont réalisées de façon plus performante avec Awk. Je signale toutefois que pour des opérations plus complexes, qui nécessitent par exemple de manipuler simultènement plusieurs fichiers, le langage Perl est mieux adapté.

2. Vous devrez tout d'abord télécharger bash.exe, gawk.exe, wc.exe et sort.exe, et les copier dans le dossier "c:/windows". Ouvrez une fenêtre de commandes MS-DOS, puis déplacez vous dans le répertoire contenant le fichier Graphemes.txt (c'est à dire, s'il se trouve dans c:/lexique, tapez "cd \lexique"); finalement, tapez 'bash'. Vous êtes alors prêts entrer les commandes fournies dans les exemples.

Bash permet la complétion des noms de fichiers par appui sur la touche TAB, ne pose pas de limite sur la taille d'une ligne de commande, et surtout, contrairement à DOS, permet d'entrer des arguments utilisant les simples ou double quotes.

Signalons tout de même un problème sous Windows (ca aurait été surprenant qu'il n'y en ait pas...): la dernière fois que nous avons travaillé sous Windows (il y a quelques années de cela...), les caractères accentués n'étaient pas gérés

La table ‘Graphèmes’ est contenue dans le fichier `Graphemes.txt`. Il s’agit d’un “gros” fichier, puisqu’il fait 14Mo. Pour accélérer les recherches, la première chose qu’il est conseillé de faire est de supprimer les colonnes qui ne sont pas utiles pour nos besoins. Par exemple, pour sélectionner les colonnes 1 (graph), 2 (phon) et 8 (frantfreq):

```
gawk -F'\t' '{OFS="\t"; print $1,$2,$8}' Graphemes.txt >mybase.txt
```

Le fichier résultant, `mybase.txt`, fait moins de 3 Mo... soit moins du quart de la taille originale ; les recherches seront nettement plus rapides. Vous pouvez adapter cet exemple : pour sélectionner, par exemple, les colonnes 2 et 10, il suffirait d’utiliser `print $2,$10`.[3]

Recherches par pattern

Cherchons, par exemple, les mots qui contiennent la suite de caractères ‘alle’ (p.ex. “footballeur”); pour cela, il faut entrer la commande :

```
gawk '$1~/alle/' mybase.txt
```

De nombreuses lignes sont affichées (120 précisement). Vous pouvez sauver ces résultats dans un fichier, par exemple `mots1.txt`, en utilisant le symbole de redirection ‘>’, avec la commande :

```
gawk '$1~/alle/' mybase.txt >mots1.txt
```

Il suffit ensuite d’ouvrir le fichier `mots1.txt` pour voir les résultats (sous Linux, je recommande d’utiliser le visualiseur de fichier “less”).

Le texte inclu entre les deux ‘/’ (ici ‘alle’) est appelé une expression régulière, ou encore un “pattern”. L’expression `$1~/pattern/`, signifie “le contenu de la première colonne est-il “conforme” au pattern ?”. Gawk parcourt `mybase.txt`, ligne par ligne, teste cette condition pour chaque ligne, et si elle est vérifiée, imprime la ligne entière.

Dans le cas précédent, tous les mots *contenant* ‘alle’ étaient affichés. Si l’on avait voulu les mots *commençant* par ‘alle’ (p.ex. “allemand”), alors on aurait entré :

```
gawk '$1~/^alle/' mybase.txt
```

correctement : des codes différents sont utilisés sous DOS et sous Windows... Cela pose des problèmes si l’on veut rechercher sous DOS un mot contenant, par exemple, le caractère ‘é’ dans un fichier utilisant l’encodage Windows. Plaignez-vous à Microsoft si c’est toujours le cas!

3. L’option `-F='\t'` et la commande `OFS="\t"` indiquent que les colonnes sont séparées par le caractère de tabulation (TAB). Cela permet de gérer les colonnes éventuellement vides.

Le signe ^ correspond, en quelque sorte, à “début de mot”. Pour obtenir les mots *finissant* par ‘alle’ (p.ex. emballe) :

```
gawk '$1~/alle$/' mybase.txt
```

Le signe $ signifie “fin de mot”. On peut combiner les deux signes ; ainsi la commande suivant trouve le mot ‘aller’ :

```
gawk '$1~/^aller$/' mybase.txt
```

Par exemple, si l’on veut extraire de Grapheme.txt uniquement les lignes qui correspondent à des NOMs (et uniquement à des NOMs) :

```
gawk '$3~/^NOM$/' Graphemes.txt
```

Dans un pattern, le symbole “point” désigne “n’importe quel caractère”. C’est particulièrement utile pour résoudre des mots croisés : par exemple, pour trouver les mots de cinq lettres dont la seconde est ‘b’ et qui finissent par ’e’:

```
gawk '$1~/^.b..e$/' ~pallier/chris01/dicos/mybase.txt
```

Cela produit:

```
aboie   abwa    1.23
abuse   abyz    1.58
abyme   abim    0.06
abîme   abim    13.23
ibère   ibER    0.06
obole   ObOl    0.71
obvie   Obvi    0.19
obère   ObER    0.13
obèse   ObEz    2.45
obéie   Obei    0.19
sbire   sbiR    0.06
ébène   ebEn    4.00
```

Les patterns offrent bien d'autres possibilités. Voici quelques exemples :

Pattern	Mots détectés
`^al`	commence par 'al'
`ba.l`	contient 'ba', puis une lettre quelconque, puis 'l'
`ba..l`	contient 'ba', puis deux lettres quelconques, puis 'l'
`ba.*l`	contient 'ba', éventuellement une suite de lettres quelconques, puis 'l'
`[aeiou]$`	fini par une voyelle
`^[ptkbdg][lr]`	commence par une plosive, suivi d'un 'l' ou d'un 'r'
`[^abc]`	ne contenant ni 'a', ni 'b', ni 'c'
`bo+l`	contient 'b', 1 ou plusieurs 'o', suivi de 'l'
`^b.*l$`	commence par 'b' et fini par 'l'

Les documentations de Awk (par exemple, celle disponible par 'info gawk' sous linux), ainsi que de nombreux documents accessibles sur le Net, expliquent les règles de formation des patterns ("regular expressions" en anglais).

Plutôt que de rechercher des patterns, on peut vouloir chercher les mots dont la fréquence est supérieure à un certain seuil, par exemple 1000 par million:

```
gawk '$3>1000' mybase.txt
```

ou encore, pour les mots ayant une fréquence comprise entre 100 et 200 :

```
gawk '$3>100 && $3<200' mybase.txt
```

L'opérateur `&&` ("et") permet de combiner plusieurs critères. Voici un autre exemple : pour chercher les mots de 4 lettres ou plus, commençant par p, t ou k, et de fréquence supérieure à 100, vous pouvez taper :

```
gawk '$3>100 && $1~/^[ptk]/ && length($1)>=4' mybase.txt
```

Il existe également un opérateur "ou" (`||`). Ainsi, pour chercher les mots de 4 ou 6 lettres :

```
gawk 'length($1)==4 || length($1)==6' mybase.txt
```

Revenons à Graphèmes.txt : pour en extraire les noms monosyllabiques de fréquences supérieure à 10, on peut entrer :

```
gawk '$3~/NOM/ && $16==1 && $3>=10' Graphemes.txt
```

Calculs statistiques

Plutôt que de simplement rechercher des mots, on peut vouloir calculer des statistiques. Par exemple, pour calculer le nombre de graphèmes comprenant 8 lettres :

```
gawk 'length($1)==8' mybase.txt | wc
```

wc ("word count") est un programme externe qui renvoit le nombre de lignes, mots et caractères contenus dans un fichier. Donc le premier nombre qu'il renvoit indique le nombre d'items trouvés dans mybase.txt. En fait, on aurait pu se passer de wc et utiliser directement la construction "END" de gawk:

```
gawk 'length($1)==8' {n++} END {print n}' Graphemes.txt
```

Ce programme awk signifie "incrémenter la variable 'n' pour chaque ligne où la colonne 1 contient 8 caractères et, à la fin, afficher 'n"'. (La commande qui suit le mot clé "END" est exécutée quand la fin du fichier d'entré est atteinte). Cette construction permet de faire des calculs complexes : par exemple, pour calculer le nombre de graphèmes de 4 lettres, ainsi que leur fréquence moyenne :

```
gawk 'length($1)==4 {n++;f+=$3} END { print n,f/n }' mybase.txt
```

On peut raffiner, en affichant ces statistiques pour chacune des longueurs:

```
gawk '{l=length($1);n[l]++;f[l]+=$3}
      END { for (l in n) print l,n[l],f[l]/n[l]}' mybase.txt | sort -n
```

La commande `sort`, placée à la fin de la ligne, permet de trier la sortie de la commande awk. Par exemple, la ligne suivante affiche les graphèmes de fréquence supérieure à 1000, triés par ordre alphabétique :

```
gawk '$3>1000' mybase.txt | sort
```

La ligne suivante fait la même chose, sauf que la sortie de gawk est triée par ordre décroissant des fréquences (qui sont dans la troisième colonne) :

```
gawk '$3>1000' mybase.txt | sort -nr -k3
```

Consultez la documentation de sort (par un "man sort" sous Linux) pour obtenir plus de détails sur les options de tri.

Voici un exemple un peu plus complexe, qui fait appel aux tableaux associatifs de Awk [4]. Il calcule et affiche les nombres d'occurence des phonèmes dans la liste des formes phonémiques :

```
gawk '{for (i=1;i<length($2);i++) { n[substr($2,i,1)]++ } }
       END{ for (i in n) { print i,n[i]} }' mybase.txt
```

Plus on veut faire des choses compliquées, plus la ligne de commandes s'allonge. Il est possible d'écrire les commandes awk dans un fichier texte, et de passer à gawk le nom du fichier à exécuter. Ouvrez par exemple un éditeur de fichier ascii (e.g. notepad), et entrez les lignes suivantes dans un fichier que vous sauverez sous le nom '`test.awk`':

```
{ for (i=1;i<length($2);i++) { n[substr($2,i,1)]++ }
}
END{ for (i in n) {print i,n[i]}}
```

Il vous suffira ensuite de taper '`gawk -f test.awk mybase.txt`' pour effectuer la recherche précedente: le nom de fichier de commandes awk est spécifié par l'intermédiare de l'option -f.

Calcul des voisins

Le script awk suivant trouve tous les mots qui diffèrent d'un mot donné par une lettre:

```
# détection des voisins orthographiques
BEGIN {
    if (mot=="") { # la variable mot doit etre definie sur la ligne de commande
        printf ("passer le mot avec l'option -vmot=...");
    }
    l=length(mot)
}

length($1)==l {
 d=0;
 for (i=1; i<=l;i++) {
  if (substr($1,i,1)!=substr(mot,i,1)) d++;
  if (d>1) next;
 }
 if (d) print $1;
}
```

4. Un tableau associatif est un tableau dont les indices peuvent être des chaines de caractères quelconques, plutôt que de simples entiers, comme dans les langages de programmation classiques. C'est un mécanisme très puissant qu'il faut essayer d'apprendre lors qu'on s'initie à Awk

Si vous sauvez ce script dans un fichier `voisins.awk`, vous pourrez obtenir les voisins orthographiques des mots ‘lait’ et ‘chien’ (par exemple), en tapant :

```
gawk -f voisins.awk -vmot=lait mybase.txt
gawk -f voisins.awk -vmot=chien mybase.txt
```

Syllabation de transcriptions phonétiques

Le script disponible à l'addresse http://www.pallier.org/ressources/syllabif/syllabation.awk permet de détecter les frontières de syllabes dans une transcription phonétique.

Il est décrit dans http://www.pallier.org/ressources/syllabif/syllabation.pdf