ORDRE LEXICOGRAPHIQUE EN FRANÇAIS

ALGORITHME UCA, TRI D'INDEX

Résumé

Doit-on classer *côte* avant ou après *coté* ?
Existe-t-il un algorithme informatique permettant de classer les mots et locutions françaises dans le même ordre que les dictionnaires classiques ?
Quel programme choisir pour créer un index en LaTeX ?
C'est à ces questions qu'essaie de répondre ce court mémoire.

Ordre lexicographique en français

L'ordre de classement des mots et locutions dans un dictionnaire français semble naturel : on triera par exemple sans hésitation[1] *acompte* < *Açores* < *à-côté* < *à-coup* < *a priori*.

L'ordre alphabétique est connu dans ses grandes lignes mais les détails le sont moins. Rappelons quelques règles de base :

- les caractères accentués (*à*, *é*, *ô*) et les diacritiques (*ç*) se classent avec leur lettre de base (*a*, *e*, *c* respectivement), en fait la lettre de base précède immédiatement sa variante accentuée, ainsi *cote* < *coté* et *côte* < *côté* ;
- l'ordre des accents en français est identique quelle que soit la lettre de base, nous le donnons uniquement pour le *e*[2] : *e* < *é* < *è* < *ê* < *ë* ;
- les ligatures *æ* et *œ* sont traitées comme deux lettres *ae* et *oe* :
  exact < *ex æquo* < *exagération*
  coefficient < *cœur* < *coexistence*
- la minuscule précède la majuscule (*a* < *A*) ;
- enfin les tirets (et avec eux les espaces, apostrophes, etc.) sont ignorés.

Une chose est claire : un tri binaire comparant les codes des caractères est totalement exclu quel que soit le codage envisagé (iso-latin-9, unicode, etc.). Rappelons en effet que le codage ASCII, base de tous les autres, place toutes les capitales avant les minuscules, on aurait donc *Zoo* avant *abat*… De même, les codes des lettres de base sont inférieurs à 127 tandis ceux des caractères accentués sont supérieurs à 128, ce qui conduirait à placer *coton* entre *cote* et *coté* !

Certains choix sont moins évidents, notamment le classement en présence d'accents multiples ; ainsi, les dictionnaires classiques (Larousse, Littré, Robert,…) ont toujours classé :
cote < *côte* < *coté* < *côté*
élève < *élevé*
gène < *gêne* < *gêné*
pèche < *pêche* < *péché* < *pêché*
relève < *relevé*

Les raisons linguistiques ayant conduit à ces choix sont disparates et pas toujours écrites, mais il semble y avoir un consensus implicite : on ne note pas de divergences entre Larousse, Littré et Robert par exemple.

Une règle simple aide à départager les substantifs homographes[3] : le féminin précède le masculin, ainsi (la) *relève* doit être classée avant (le) *relevé*, (la) *pêche* avant (le) *péché*[4].

1. Dans tout ce document le signe < s'entend au sens de la relation d'ordre lexicographique, a < b signifiant *a* vient avant *b*.
2. Car c'est la lettre qui admet le plus de variantes accentuées en français.
3. Mots identiques aux accents et diacritiques près, exemple *cote*, *côte*, *coté*, *côté*.
4. Noter que cette règle ne s'applique que si l'ordre des accents rappelé ci-dessous n'a pas permis le classement, ainsi (le) *gène* précède (la) *gêne* car *è* < *ê*.

Trouver un algorithme aboutissant automatiquement au classement traditionnel des mots et locutions françaises n'a pas été simple. Le problème est maintenant résolu grâce à l'algorithme UCA et aux contributions d'une équipe canadienne autour d'Alain LaBonté.

De nos jours, les recherches dans les bases de données imposent que s'applique un ordre unique connu de tous les utilisateurs. Un classement reposant sur un algorithme est un gage de stabilité et de reproductibilité. Mais le classement algorithmique se doit d'être compatible avec le classement historique des dictionnaires classiques de la langue concernée.

Notons que l'ordre lexicographique est très dépendant de la langue : un francophone s'attend à trouver *œdème* sous la lettre *O* tandis qu'un Suédois cherchera le mot *öde* (désert) à la lettre Ö après le *Z*. Les Allemands ont deux modes de classement, DIN 5007-1 (cas général : *ä* traité comme *a*) et DIN 5007-2 (listes de noms propres : *ä* traité comme *ae*).

Une dernière précision : il est d'usage de classer les locutions étrangères comme celles de la langue principale du document. L'index d'un document en français placera donc *föhn* entre *fœtus* et *foi* et le mot suédois *öde* à la lettre *O*.

Principes de l'algorithme UCA[5]

Un tri à trois niveaux

Vouloir classer ensemble les homographes amène à proposer un premier niveau de tri où tous les accents sont supprimés. L'usage étant de ne pas tenir compte (dans un premier temps) de la casse, on négligera également celle-ci au premier niveau.

À l'issue de cette première phase, les homographes sont classés ex æquo : ils pourront être départagés lors d'une seconde phase de tri. Une troisième phase permettra de départager majuscules et minuscules : *à* et *À* sont classés ex æquo aux niveaux un et deux.

L'algorithme UCA reprend ce principe. Nous le décrivons uniquement pour les langues latines[6]. L'idée de base est d'associer à chaque chaîne de caractères une clé de tri[7] qui pourra ensuite être traitée numériquement.

À chaque caractère sont associés trois « poids » (primaire, secondaire, tertiaire) qui déterminent l'ordre de classement aux niveaux 1, 2 et 3 respectivement.

Le poids primaire est pris en compte au premier niveau de tri (sans accents ni diacritiques). Ainsi un *a* ou un *à* auront le même poids primaire, plus faible qu'un *b*. La minuscule et la majuscule correspondante ont toujours le même poids primaire ce qui assure de les classer ex æquo au premier niveau.

La liste de ces « poids » est donnée dans une table appelée DUCET.

La prise en compte des poids se fait dans le sens de la lecture, donc de gauche à droite en français, et la comparaison s'arrête à la première différence trouvée.

Prenons un exemple concret : comparons les chaînes de caractères *Cote*, *côte*, *coté* et *côté*. La table DUCET donne les « poids » suivants[8] (en base 10) :

5. L'algorithme UCA est décrit en détail dans [1].

6. UCA est suffisamment flexible pour traiter toutes les langues alphabétiques ou non, arabe, chinois, langues indiennes comprises.

7. Les utilisateurs avisés de `makeindex` ont recours à une telle clé lorsqu'ils veulent classer correctement un mot comme *été* : ils codent `\index{ete@été}` pour associer la clé `ete` au mot *été*.

8. En réalité, les lettres accentuées sont prises en compte comme deux caractères successifs (ou plus), la lettre de base suivie de son (ou ses) accents(s). Ainsi dans la table DUCET, l'accent aigu a pour poids (0,39,2), le grave (0,37,2). Sachant que les poids nuls sont ignorés à chaque niveau de tri, la présentation simplifiée que nous donnons ici est correcte, mais uniquement pour les langues n'ayant aucun caractère qui porte plusieurs

	Primaire	Secondaire	Tertiaire
C	7617	32	8
c	7617	32	2
o	7972	32	2
ô	7972	39	2
t	8157	32	2
e	7665	32	2
é	7665	37	2

On constate que les poids primaires sont identiques pour *C* et *c*, ainsi que pour les voyelles avec ou sans accent (*o* et *ô*, *e* et *é*). Les poids des *c*, *e*, *o* et *t* sont bien en ordre croissant. Au niveau primaire, les quatre chaînes ont la même clé de tri (7617,7972,8157,7665) : elles seront donc classées ex æquo.

En revanche, un mot comme *coopérer* serait classé avant elles au niveau primaire (au bout de seulement trois comparaisons), le second *o* ayant un poids inférieur au *t*.

Autre exemple. Les deux premières lettres des mots *ça* et *car* ont le même poids primaire : c'est la longueur qui fait la différence dès le premier niveau, *ça* < *car*, parce que l'absence de troisième caractère dans *ça* est interprétée comme un caractère vide de poids 0.

Au niveau secondaire, le mot *Cote* sera évidemment classé en premier, sa clé de tri étant (32,32,32,32). Comparons ensuite *côte* et *coté* : en lisant de gauche à droite, le *o* a un poids inférieur à celui de *ô*, donc *coté* devrait être classé avant *côte* ; malheureusement ce classement contredirait l'usage… Une équipe canadienne, autour d'Alain LaBonté, a proposé d'inverser l'ordre de lecture pour classer les accents en français (donc uniquement au niveau deux). De fait, le résultat est alors conforme au classement traditionnel des dictionnaires français… pas seulement sur cet exemple ! Alain Rey lui-même a donné acte à Alain Labonté que sa méthode produisait un classement en tout point conforme à l'usage en français.

Vérifions-le sur notre exemple. Si on prend en compte les accents de droite à gauche, c.-à-d. à partir de la fin du mot en remontant :

— dès la première comparaison, les *é* finaux de *coté* et *côté* (poids 37) conduisent à les placer après *côte* (poids 32) ;
— il reste à comparer *côté* et *coté* : la troisième comparaison impose *coté* < *côté* (32 contre 39).

Le classement final est donc *Cote* < *côte* < *coté* < *côté*.

Dans la table DUCET, le poids secondaire des accents est propre à l'accent et indépendant de la lettre qui le porte. Il conduit au classement suivant qui convient parfaitement au français : *e* < *é* < *è* < *ê* < *ë* (poids respectifs 32, 36, 37, 39, 43).

Le consortium Unicode, garant de l'algorithme UCA, a officiellement entériné la possibilité d'inverser l'ordre de lecture pour chaque niveau. À ce jour, cette possibilité n'est utilisée qu'au niveau deux et uniquement pour le français.

Le troisième niveau permet par exemple de comparer *Cote* et *cote*. Les capitales ont un poids tertiaire de 8 contre 2 pour les minuscules, ce qui place *Cote* après *cote*, c'est effectivement l'ordre adopté en français. Une option permet d'inverser cet ordre pour placer les majuscules avant les minuscules, comme c'est l'usage en allemand par exemple.

accents. C'est le cas du français, mais pas du vietnamien par exemple.

Un quatrième niveau (optionnel)

Il reste à préciser comment classer les chaînes de caractères comportant des espaces, traits d'union, apostrophes, etc.

Dans l'algorithme UCA de base à trois niveaux, ces caractères sont traités de la même façon que les lettres, en fonction de leurs poids dans la table DUCET :

	code	Primaire	Secondaire	Tertiaire
`<espace >`	U+0020	521	32	2
`<tiret ->`	U+002D	525	32	2
`<apost. '>`	U+0027	785	32	2
`<apost. ’>`	U+2019	787	32	2

Leur poids primaire est nettement inférieur à celui des autres caractères, ainsi en anglais le tri UCA à trois niveaux classera *stand* < *stand by* < *stand for* < *stand up* < *standard*, l'espace ayant un poids primaire inférieur à celui du *a*. Le résultat est satisfaisant dans ce cas mais l'est moins si on considère les locutions *death*, *de-escalate* et *de facto*, qui seront classées *de facto* < *de-escalate* < *death*, l'ordre attendu étant plutôt *death* < *de-escalate* < *de facto*.

L'algorithme UCA propose plusieurs options permettant d'ignorer les caractères non alphabétiques (espaces, tirets et apostrophes ainsi que d'autres) aux trois premiers niveaux de tri. Un quatrième niveau de tri est alors nécessaire pour départager *tire-bouchon* et *tirebouchon* par exemple.

La table DUCET donne pour chaque caractère, en plus des trois poids mentionnés ci-dessus, une quatrième valeur binaire dite de « type » : soit « type lettre », soit « type autre ». Lors des trois premiers niveaux de tri, les caractères de type « lettre » sont toujours pris en compte, tandis que ceux de type « autre », aussi appelés « à poids variable » (*Variable weight*), peuvent être ignorés : en activant l'une des options `[Alternate Shifted]` ou `[Alternate Shift-Trimmed]`, les caractères de type « autre » voient leurs poids annulés aux niveaux un à trois[9] et sont crédités au niveau quatre d'un poids égal à leur ancien poids primaire de base. Ainsi, si l'une des options `[Alternate Shifted]` ou `[Alternate Shift-Trimmed]` est activée, le tiret U+002D est pris en compte avec les poids (0, 0, 0, 525), l'espace avec (0, 0, 0, 521), etc. L'option `[Alternate Shift-Trimmed]` attribue, au niveau quatre, un poids *maximal* aux caractères de type « lettre » (0xFFFF en hexadécimal soit 65535), tandis que `[Alternate Shifted]` leur attribue un poids *minimal* (inférieur à celui de tous les caractères de type « autre »).

En pratique, l'option `[Alternate Shift-Trimmed]` classe *au quatrième niveau*[10] les caractères de type « lettre » *avant* (*coop* < *co-op*, *LaFayette* < *La Fayette*), tandis que l'option `[Alternate Shifted]` les place *après* les espaces, tirets, apostrophes et autres (*co-op* < *coop*, *La Fayette* < *LaFayette*).

Exemples :

- Le tri de base donne :
  - *ça et là* < *cæsium* ;
  - *l'âme* < *lame* < *lamé* ;
  - *tire-clou* < *tire-d'aile* < *tire-dent* < *tirebouchon* < *tirefond*.
- Les options `[Alternate Shifted]` et `[Alternate Shift-Trimmed]` classent toutes deux :
  - *cæsium* < *ça et là* < *cafard* ;

9. Rappel : les poids nuls sont toujours supprimés de la clé de tri.
10. Il est bien rare qu'il faille aller jusque-là pour départager deux chaînes de caractères !

— *lame* < *l'âme* < *lamé*[11] ;
— *tirebouchon* < *tire-clou* < *tire-d'aile* < *tire-dent* < *tirefond*.

Les options `[Alternate Shift-Trimmed]` ou, à défaut, `[Alternate Shifted]`, sont recommandées pour le français.

Adaptations de l'algorithme UCA aux *locales*

Nous avons vu que les poids attribués par la table DUCET conviennent au traitement du français. Pour d'autres langues, il peut être nécessaire d'adapter cette table en modifiant les poids relatifs de certains caractères. Cette opération s'appelle *tailoring* en anglais, voir [1] section 8, ou [2] section 12.6.3.

La syntaxe à utiliser est assez simple : pour définir un ordre relatif pour les poids primaires on utilise le signe <, << pour les poids secondaires et <<< pour les poids tertiaires. Un locuteur d'une langue scandinave désirant classer dans l'ordre *Æ*, *Ø* et *Å* après *Z* imposera la règle `tailoring("&Z<Æ<Ø<Å")`. En espagnol, pour classer *llano* après *luz*, il suffit d'ajouter la règle `tailoring("&l<ll<<<lL<<<LL")`.

Mise en œuvre informatique : exemple de lua-uca

L'algorithme UCA a été codé d'abord en Java et plus récemment en Lua par Michal Hoftich, lua-uca [3]. Le tri pour le français fonctionne bien depuis la version 0.1d[12] sauf pour les locutions et mots composés (« ça et là », « tire-bouchon »), qui nécessitent le recours au quatrième niveau de tri de l'algorithme UCA. En effet, la version Lua de Michal Hoftich se limite aux trois premiers niveaux.

Le document [4] propose une liste de locutions francophones et indique l'ordre à obtenir. C'est un excellent test de torture pour les programmes de tri, les exemples qui suivent sont basés sur cette liste.

Voici un fichier `sort-list-fr.lua` à compiler avec LuaTEX, qui utilise l'algorithme de lua-uca pour trier la liste de mots contenus dans la table `t`. Cette liste est construite à partir de la liste canadienne (locutions et mots composés ou étrangers exclus), avec quelques ajouts personnels.

Exemple 1

```
#!/usr/bin/env texlua

local t = {"CÔTÉ", "cote", "Côté", "COTÉ", "côte", "COTE",
           "côté", "Coté", "coté", "Cote", "CÔTE", "Côte",
           "lésé", "péché", "bohème", "gêné", "pêche",
           "cæsium", "pêcher", "révèle", "pécher", "révélé",
           "Bohême", "relève", "PÉCHÉ", "maçon", "relevé",
           "Élève", "gêne", "élevé", "MÂCON", "gène",
           "Bohémien", "caennais", "lèse", "coexistence",
           "cœur", "coefficient", "cafard", "CŒUR", "CÆSIUM"}

kpse.set_program_name "luatex"
local ducet = require "lua-uca.lua-uca-ducet"
```

11. Avec lecture des accents à rebours.
12. NDLR : Cette version étant récente, vous devrez probablement mettre à jour manuellement le package lua-uca pour en bénéficier (notamment si vous souhaitez tester l'exemple ci-dessous), en utilisant par exemple `tlmgr`.

```
local collator = require "lua-uca.lua-uca-collator"
local languages = require "lua-uca.lua-uca-languages"

local collator_obj = collator.new(ducet)
-- load French rules
collator_obj = languages.fr_backward_accents(collator_obj)

table.sort(t, function(a,b)
  return collator_obj:compare_strings(a,b)
end)

for _, v in ipairs(t) do
  print(v .. ",")
end
```

L'exécution donne ceci :

```
$ luatex "sort-list-fr.lua"

bohème, Bohême, Bohémien, caennais, cæsium, CÆSIUM,
cafard, coefficient, cœur, CŒUR, coexistence, cote,
Cote, COTE, côte, Côte, CÔTE, coté, Coté, COTÉ,
côté, Côté, CÔTÉ, Élève, élevé, gène, gêne, gêné,
lèse, lésé, MÂCON, maçon, pêche, péché, PÉCHÉ,
pécher, pêcher, relève, relevé, révèle, révélé,
```

Le résultat est un sans-faute.

Un second test porte sur le classement des locutions et mots composés. Le quatrième niveau de tri n'étant pas implémenté dans lua-uca, le résultat n'est pas fameux. On reprend le même script avec la table suivante :

Exemple 2

```
local t = { "vice-président", "Ça", "vice versa", "C.A.F.",
            "tire-dent", "L'Haÿ-les-Roses", "tire-clou",
            "caennais", "co-op", "lame", "Mc Arthur","colon",
            "tirefond", "l'âme", "Canon", "McArthur", "lamé",
            "Mc Mahon", "tire-d'aile", "çà et là", "tirebouchon",
            "MacArthur", "lésé", "maçon", "cæsium", "coop", }
```

Avec cette nouvelle table la compilation donne :

```
$ luatex "sort-list-fr.lua"

C.A.F., Ça, çà et là, caennais, cæsium, Canon,
co-op, colon, coop, l'âme, L'Haÿ-les-Roses, lame, lamé, lésé,
MacArthur, maçon, Mc Arthur, Mc Mahon, McArthur,
tire-clou, tire-d'aile, tire-dent, tirebouchon, tirefond,
vice versa, vice-président,
```

Le bon ordre serait :

```
$ luatex "sort-list-fr.lua"
```

```
caennais, cæsium, çà et là, C.A.F., Canon, colon,
coop, co-op, lame, l'âme, lamé, lésé, L'Haÿ-les-Roses,
MacArthur, maçon, McArthur, Mc Arthur, Mc Mahon,
tirebouchon, tire-clou, tire-d'aile, tire-dent, tirefond
vice-président, vice versa,
```

En effet, les espaces, apostrophes, tirets et points devraient être ignorés aux trois premiers niveaux, le quatrième niveau départageant *coop* et *co-op* ainsi que *McArthur* et *Mc Arthur*.

Comparaison des programmes de création d'index pour LaTeX

Le cas le plus fréquent où se manifeste le besoin de trier une liste de mots est celui de la création d'un index.

makeindex est le programme le plus ancien, il est limité au tri des caractères ASCII, ce qui le disqualifie pour le français, à moins de créer des clés ASCII pour les mots accentués comme ceci : `\index{ete@été}`. Noter que le classement correct des mots *relève* et *relevé* suppose la création des clés suivantes :

- `\index{releve1@relève}`
- et `\index{releve2@relevé}`

Cela implique que l'auteur connaisse parfaitement les règles de tri en français.

xindy disponible depuis une vingtaine d'année est adapté aux caractères accentués, mais il a été conçu avant le développement d'Unicode essentiellement pour des codages 8-bits (le *é* est codé en LATIN1 ou en LATIN9 sur un seul octet, alors qu'il est codé sur deux octets en UTF-8). Aujourd'hui, les sources LaTeX devraient *tous* être codés en UTF-8 (codage par défaut pour pdfLaTeX et obligatoire pour LuaLaTeX et XeLaTeX), si bien que le recours au programme `xindy` n'est plus pertinent. De plus, `xindy` ne respecte pas complètement l'ordre lexicographique des dictionnaires français : il classe par exemple *relève* (féminin) après *relevé* (masculin).

xindex[5] est écrit en Lua et adapté au codage UTF-8 ; depuis la version 0.60 (mai 2024), le tri est fait par lua-uca (option par défaut) et donne des résultats satisfaisants en français, sauf bien sûr pour les mots composés comme indiqué ci-dessus.

upmendex [6] est un autre programme adapté au codage UTF-8, il est écrit en C, utilise la bibliothèque UCA d'origine et non le portage en Lua de Michal Hoftig. C'est le programme de tri d'index qui donne actuellement les meilleurs résultats en français.

Voyons maintenant comment utiliser `xindex` et `upmendex` sur un exemple compact mais exigeant en termes de tri. Le source LaTeX est le suivant :

Exemple 3

```
\documentclass[french]{article}
\usepackage{babel}
% Pour xindex
\usepackage{xindex}\makeindex
% Pour upmendex
%\usepackage{makeidx}\makeindex

\newcommand*{\IND}[1]{\index{#1}#1\par}
```

```
\setlength{\parindent}{0pt}
\begin{document}
\thispagestyle{empty}

\IND{CÔTÉ} \IND{cote} \IND{çà et là} \IND{Côté} \IND{Bohémien}
\IND{L'Haÿ-les-Roses} \IND{côte} \IND{COTE} \IND{tire-clou}
\IND{côté} \IND{Coté} \IND{C.A.F.} \IND{coté} \IND{l'âme}
\IND{Cote} \IND{MacArthur} \IND{élevé} \IND{CÆSIUM} \IND{lamé}
\IND{CÔTE} \IND{Mc Arthur} \IND{tirebouchon} \IND{Côte}
\IND{lésé} \IND{MÂCON} \IND{co-op} \IND{péché} \IND{tirefond}
\IND{bohème} \IND{lame} \IND{McArthur} \IND{gêné} \IND{pêche}
\IND{cæsium} \IND{tire-dent} \IND{Mc Mahon} \IND{pêcher}
\IND{révèle} \IND{cafard} \IND{relevé} \IND{Bohême} \IND{pécher}
\IND{relève} \IND{PÉCHÉ} \IND{vice versa} \IND{maçon}
\IND{coop} \IND{Élève} \IND{gêne} \IND{CŒUR} \IND{vice-président}
\IND{gène} \IND{COTÉ} \IND{caennais} \IND{lèse} \IND{coexistence}
\IND{cœur} \IND{coefficient} \IND{tire-d'aile \IND{révélé}

\newpage
\IND{coté} \IND{cœur}  \IND{tire-d'aile} \IND{péché}
\newpage
\IND{coté} \IND{vice versa} \IND{coefficient}

\printindex
\end{document}
```

Utilisation de `xindex`

À l'issue d'une première compilation créant le fichier `.idx`, on exécute la commande :
`xindex -l fr -i -c let-gut test-french`
qui crée le fichier `.ind`.

L'option `-l fr` (*indispensable*) impose la langue française pour le tri.

L'option `-i` (`ignore-spaces`) améliore le classement de *ça et là* qui est correctement classé entre *CÆSIUM* et *cafard* ; sans cette option, il serait classé en deuxième position juste après *C.A.F.*. Cette option n'a aucune influence sur le classement des mots composés comme *tire-bouchon* ou *l'âme*.

L'option `-c let-gut` charge le fichier de configuration `xindex-letgut.lua` qui influe sur la présentation de l'index à la manière d'un fichier `.ist` pour `makeindex` ou `upmendex`.

Une seconde compilation produit l'index de gauche, figure 1 page 10.

Utilisation de `upmendex`

`upmendex` est (presque) totalement compatible avec `makeindex`, les options sont les mêmes (à l'exception de `-g` qui concerne l'allemand sous `makeindex` et le japonais sous `upmendex`). L'option `-s` permet de charger un fichier de style : noter que `upmendex` accepte plusieurs options `-s` successives, et que les fichiers de style conçus pour `makeindex` fonctionnent à l'identique sous `upmendex`.

Le choix de la langue de base pour le tri se fait dans un fichier de style `.ist`. En voici un exemple adapté au français, appelons-le `french.ist` :

Exemple 4

```
% Règles de tri pour le français :
icu_attributes "french-collation:on strength:tertiary alternate:
    shifted"
```

Pour produire un index d'aspect semblable au précédent on ajoute le fichier de configuration `myindex.ist` (syntaxe de `makeindex`) :

Exemple 5

```
% insertion avant la lettre
heading_prefix "{\\bfseries "
% ajout après la lettre
heading_suffix "\\hfil}\\nopagebreak\n"
% activation impression lettre
headings_flag  1

% n° de page
delim_0        ", "
delim_1        ", "
delim_2        ", "
```

La commande à exécuter pour produire le fichier `.ind` est la suivante :
`upmendex -s french.ist -s myindex.ist test-french`
Une description plus complète de upmendex est donnée dans *The LateX Companion 3e éd.* vol. II, p. 364-370.

Une seconde compilation produit l'index de droite de la figure 1 page suivante.

Comparaison des deux index

xindex comme upmendex trient parfaitement les caractères accentués de notre langue, aucune inversion n'est à déplorer dans la liste canadienne.

La différence est perceptible sur les locutions et mots composés.

— L'option `-i` de xindex fonctionne bien pour les locutions ; les espaces étant ignorés, *ça et là* est bien classé entre *CÆSIUM* et *cafard* alors que sans cette option il serait en deuxième position juste derrière *C.A.F.* (le point est classé avant toute lettre). En revanche, le classement des mots composés (contenant une apostrophe ou un tiret) laisse à désirer : *l'âme* et *L'Haÿ-les-Roses* sont classés avant *lame*, de même *tire-clou* est avant *tirebouchon*, etc. Ceci résulte de la non prise en compte du quatrième niveau de tri dans l'implémentation lua-uca.

— upmendex donne un résultat parfaitement conforme aux attentes : *C.A.F.* est classé juste avant *cafard* et non plus en tête de liste, et les mots commençant par *tire* avec ou sans trait d'union sont dans le bon ordre.

Index

B
bohème, 1
Bohême, 1
Bohémien, 1

C
C.A.F., 1
caennais, 1
cæsium, 1
CÆSIUM, 1
çà et là, 1
cafard, 1
co-op, 1
coefficient, 1, 3
cœur, 1, 2
CŒUR, 1
coexistence, 1
coop, 1
cote, 1
Cote, 1
COTE, 1
côte, 1
Côte, 1
CÔTE, 1
coté, 1-3
Coté, 1
COTÉ, 1
côté, 1
Côté, 1
CÔTÉ, 1

E
Élève, 1
élevé, 1

G
gène, 1
gêne, 1
gêné, 1

L
l'âme, 1
L'Haÿ-les-Roses, 1
lame, 1
lamé, 1
lèse, 1
lésé, 1

M
MacArthur, 1
MÂCON, 1
maçon, 1
McArthur, 1
Mc Arthur, 1
Mc Mahon, 1

P
pêche, 1
péché, 1, 2
PÉCHÉ, 1
pécher, 1
pêcher, 1

R
relève, 1
relevé, 1
révèle, 1

T
tire-clou, 1
tire-d'aile, 2
tire-dent, 1
tirebouchon, 1
tirefond, 1

V
vice-président, 1
vice versa, 1, 3

Index

B
bohème, 1
Bohême, 1
Bohémien, 1

C
caennais, 1
cæsium, 1
CÆSIUM, 1
çà et là, 1
C.A.F., 1
cafard, 1
coefficient, 1, 3
cœur, 1, 2
CŒUR, 1
coexistence, 1
co-op, 1
coop, 1
cote, 1
Cote, 1
COTE, 1
côte, 1
Côte, 1
CÔTE, 1
coté, 1–3
Coté, 1
COTÉ, 1
côté, 1
Côté, 1
CÔTÉ, 1

E
Élève, 1
élevé, 1

G
gène, 1
gêne, 1
gêné, 1

L
lame, 1
l'âme, 1
lamé, 1
lèse, 1
lésé, 1
L'Haÿ-les-Roses, 1

M
MacArthur, 1
MÂCON, 1
maçon, 1
Mc Arthur, 1
McArthur, 1
Mc Mahon, 1

P
pêche, 1
péché, 1, 2
PÉCHÉ, 1
pécher, 1
pêcher, 1

R
relève, 1
relevé, 1
révèle, 1

T
tirebouchon, 1
tire-clou, 1
tire-d'aile, 2
tire-dent, 1
tirefond, 1

V
vice-président, 1
vice versa, 1, 3

Figure 1 – À gauche `xindex`, à droite `upmendex`

Conclusion

upmendex est actuellement le plus performant des programmes de création d'index, xindex vient juste derrière mais il est plus simple à utiliser et bien suffisant en pratique. Pour les départager, il a fallu recourir à une liste de mots vraiment sélective : qui a vraiment besoin de classer correctement *l'âme* ou *L'Haÿ-les-Roses* dans un index ? D'autre part, les traits d'union ont tendance à disparaître depuis la réforme de 1990 ; quant aux abréviations, on écrit plutôt *CAF* que *C.A.F.* pour *Caisse d'allocations familiales*…

La bonne nouvelle est que nous disposons *enfin* de *deux* programmes qui produisent des index conformes à l'ordre lexicographique traditionnel des dictionnaires français, les utilisateurs ont donc le choix : xindex ou upmendex.

Remerciements

Ayant toujours trouvé décevants les résultats obtenus pour le tri des index en français par makeindex et xindy, j'ai découvert il y a deux ans l'existence d'un nouveau programme de tri basé sur Unicode, lua-uca de Michal Hoftig.

Comme beaucoup, je n'avais pas d'idée précise sur le classement des mots *cote*, *côte*, *coté* et *côté* à part ce qui en était dit dans le *The LateX Companion 2e éd., trad. française* à propos de xindy. Suite à mes questions posées sur la liste Typo, Jacques André (merci Jacques !) m'a mis en contact avec Alain LaBonté qui a eu la gentillesse de répondre patiemment à mes interrogations, je l'en remercie bien vivement.

Une fois connue la liste de référence établie par les Canadiens, il restait à la confronter aux résultats produits par lua-uca. Hélas, ils n'étaient pas brillants à l'époque, lua-uca ne proposant pas le tri à l'envers des accents, mais c'est maintenant corrigé depuis la version 0.1d. Ensuite, Herbert Voß a intégré cette version de lua-uca dans la version 0.60 de xindex. Merci donc à Michal et Herbert !

Daniel Flipo

Références

[1] *Unicode Collation Algorithm*. Document officiel sur le tri Unicode. 2020. URL : https://unicode.org/reports/tr10/.

[2] Patrick Andries. *Unicode 5.0 en pratique*. Dunod, 2008. ISBN : 978-2-10051140-2.

[3] Michal Hoftig. *Package lua-uca disponible sur CTAN*. 2024. URL : https://github.com/michal-h21/lua-uca/.

[4] *Norme canadienne CAN/CSA Z243.4.1-98*. Rechercher le fichier SGQRI004.pdf. 2006. URL : https://www.tresor.gouv.qc.ca/.

[5] Herbert Voß. *Package xindex disponible sur CTAN*. 2023. URL : https://gitlab.com/hvoss49/xindex.

[6] Takuji Tanaka. *Package upmendex disponible sur CTAN*. Doc : upmendex.man1.pdf. 2023. URL : https://github.com/t-tk/upmendex-package.

[7] Frank Mittelbach et Ulrike Fischer. *The LateX Companion 3e éd.* Pearson Addison-Wesley, 2023. ISBN : 978-0-13-465894-0.

[8] Frank Mittelbach et Michel Goossens. *The LateX Companion 2e éd., trad. française*. Pearson Education France, 2005. ISBN : 978-2-7440-7133-1.