

Les expressions rationnelles au service de la préparation de copie et de la mise en pages

Introduction

Lorsqu'il s'agit de produire des livres à la composition soignée, il ne suffit pas de prêter attention à la couleur du texte, d'autant que les copies à traiter ont été souvent mal préparées : espaces insécables manquantes, styles absents, renvois à la main sans recours aux fonctions de références croisées... Demander à l'éditeur ou aux correcteurs de revoir leur copie peut être délicat. De plus, même un fichier bien préparé n'empêchera pas la survenue de multiples accidents de composition lors de la mise en pages.

Si seulement on pouvait finaliser la préparation de copie en quelques minutes et prévenir au moins certains accidents typographiques ! Et voici qu'entrent en scène les expressions rationnelles.

Une expression rationnelle¹ consiste en la description plus ou moins abstraite d'un motif de chaîne de caractères, à l'aide d'un langage informatique très concis². Il s'agira pour nous d'utiliser ce langage en vue de manipuler les textes auxquels nous travaillons à l'aide de chercher-remplacer³.

Des ouvrages et des sites entiers sont consacrés au sujet, c'est dire s'il peut être complexe. Heureusement, dans la plupart des cas qui nous concernent, les « regex » sont assez simples à écrire — un peu moins à relire, en raison de l'extrême concision du langage et de la prolifération de signes cabalistiques à mesure que croissent en complexité les motifs décrits.

Apprendre à écrire des regex réclame un certain effort. Le jeu en vaut-il la chandelle ? À chacun d'en juger, en fonction de ses propres besoins. Faute de place, il ne saurait être question ici de proposer une véritable initiation à ce langage informatique. Je me bornerai à en

1. Ou « expression régulière » (calque de *regular expression*, souvent abrégé en *regex*). Dans les quelques exemples qui suivent, on utilisera le caractère « □ » pour représenter une espace-mot ordinaire dans les regex, afin d'éviter toute ambiguïté ou problème de lecture.

2. Qui n'a en toute rigueur pas de nom.

3. Notés ensuite par commodité « CR ».

présenter les grandes lignes, avant de produire quelques exemples d'utilisation qui devraient en montrer l'intérêt.

Tour d'horizon

Nul besoin, pour commencer, d'installer un logiciel particulier : on peut écrire des expressions rationnelles dans les formulaires de CR de Writer, d'InDesign ou de son éditeur LaTeX⁴, voire d'un simple bloc-notes comme KWrite. On peut aussi les intégrer à des programmes appelés scripts ou à des macros : bien que les regex permettent déjà de décupler l'efficacité des CR manuels, seule la programmation permettra d'en déployer tout le potentiel⁵.

Une expression rationnelle décrit un ensemble de chaînes de caractères possibles ; il faut toutefois avoir conscience qu'une regex ignore les concepts de mot, de nombre, de phrase ou d'alinéa : leur langage ne connaît que *caractères*, *classes de caractères*, *quantificateurs*, *tests de position*, *groupes*, «*alternatives*», *récurtivité* et *captures* (mise en mémoire de tout ou partie de la chaîne trouvée, généralement pour l'exploiter dans l'expression utilisée pour le remplacement).

Pour prendre un exemple simple, chercher la regex `chat` permettra bien de trouver le mot *chat*, mais aussi la suite de lettres *c, h, a, t* dans *crachats* ou *chatterie*, sans trouver pour autant le mot *Chat* (sauf recherche non sensible à la casse). Si l'on cherche spécifiquement le mot *chat* au singulier, éventuellement avec grande cap dans une recherche sensible à la casse, il faudra saisir `\b[Cc]hat\b` ; et si l'on souhaite inclure les occurrences du mot au pluriel, `\b[Cc]hats?\b`. Pour chercher le mot *chat* seulement juste après *mon, ton* ou *son* (éventuellement avec cap), on utilisera `(?<=\b[MmTtSs]on_)\b[Cc]chat\b`. Nous laissons au lecteur le soin d'imaginer à quoi correspondent `\b`, `[]`, `?` et (plus dur) `(?<=)` — ces exemples ont pour seul but de donner un aperçu de ce à quoi peut ressembler une regex.

Passons à des exemples concrets pour la préparation de copie.

4. Microsoft Word propose un équivalent bancal à la syntaxe non standard, avec de sérieuses limitations — énième raison de ne pas utiliser ce logiciel.

5. Fournie gracieusement à qui le souhaite, ma macro de toilettage orthotypographique pour LibreOffice procède à des centaines de CR en quelques secondes.

Ajout d'espaces insécables

En quelques CR, les regex permettront d'ajouter des insécables⁶ entre tout nombre en chiffres et mot qui suit ; entre nom de souverain et numéro en chiffres romains ; entre titres de civilité et patronymes ; entre tout numéro de siècle et *siècle* ou l'abréviation *s.* ; avant un tiret suivi d'un signe de ponctuation, etc. Bien sûr, des extensions ou logiciels comme Grammalecte et Antidote le font aussi, mais l'on n'en dispose pas forcément en toutes circonstances.

Traitement des espaces avant/après certains signes de ponctuation

En LaTeX, il vaut mieux confier au module french de l'extension babel la gestion des espacements liés à la ponctuation : de fait, certaines polices ne disposent pas du « glyphe » correspondant au caractère U+202F (fine insécable), ou alors la chasse en est inadéquate. Il conviendra dès lors de supprimer toutes ces espaces. Un seul CR suffit :

(?<=[«»])[\s~]|\s~(?=[»»];:;!?) à remplacer par rien.

Avec d'autres logiciels, il s'agira au contraire d'ajouter les fines insécables manquantes ou de remplacer par celles-ci les espaces inadaptées (espace justifiante ou insécable non fine). On peut là aussi le faire en une fois, avec une expression plus complexe.

Empêcher les fins de page ou de ligne de type « . À »

Il est toujours agaçant de voir une belle page se terminer par « . Il » (« faut tourner la page pour lire la suite »). De toute façon, quelle qu'en soit la position sur la page, ce genre de fin de ligne n'est pas très esthétique ni confortable à la lecture. Il est facile d'y remédier avec une insécable : on écrira une regex permettant de trouver une chaîne de caractères consistant en un point ([.!?...:]), suivi d'une espace-mot, suivie d'une grande capitale et de zéro à trois caractères alphanumériques⁷, suivis d'une espace. On trouvera de la sorte tout premier mot de phrase de quatre signes ou moins (à l'exclusion du premier mot de l'alinéa, dont le traitement serait inutile), afin de substituer une insécable à l'espace qui le suivait. C'est toutefois sans compter sur la présence possible de guillemets, parenthèses ou crochets avant ce premier mot, mais je n'ai pas souhaité compliquer l'exemple.

6. Codés par ~ dans TeX.

7. Plutôt qu'alphabétiques, afin de trouver par exemple A5.

Limiter le risque de ligne à voleur

On définit d'ordinaire la ligne à voleur par un nombre de caractères inférieur à cinq⁸, ce que je ne trouve pas satisfaisant. Après tout, malgré trois lettres de plus, *effilées* chasse moins que *maman* dans la plupart des polices : or, comme le problème posé est avant tout d'ordre esthétique, c'est à mon sens la chasse de la dernière ligne de l'alinéa par rapport à la justification qu'il conviendrait de prendre en compte. Malheureusement, les regex ne peuvent mesurer la chose. On doit donc s'en tenir à un nombre de caractères, qui toutefois devrait varier selon la justification : une ligne creuse de 6 caractères pourra sembler acceptable si la justification est de 45 signes, beaucoup moins si elle est de 60.

Pour limiter le risque de ligne à voleur dès la préparation de copie, on pourrait songer à interdire la division du dernier mot de chaque alinéa (avec la commande `\bname` en LaTeX si comme moi on veut de la flexibilité, ou en l'interdisant de façon quasi systématique en attribuant une valeur élevée à `\finalhyphendemerits`). Cela ne suffira pas toujours, quand le mot en question est court, coupable ou non : *si*, *ami*, *caler*, *chats*, *revues*... On pourrait donc vouloir en plus remplacer l'espace-mot qui précède un tel mot court par une insécable permettant de souder les deux derniers mots, ce qui hélas pourrait encore ne pas suffire : `et~si..., à~\bname{caler}., le~sol`. Plutôt que de commencer par traiter le dernier mot, il vaut mieux chercher l'espace précédant une chaîne quelconque de moins de n caractères avant la fin de l'alinéa (mettons 10 pour la justif considérée), incluant donc d'éventuelles espaces, et remplacer cette espace ainsi que celles qui suivraient par des insécables. Ensuite seulement (pour des raisons pratiques, en LaTeX), on pourra identifier le dernier mot de l'alinéa et en interdire la coupure : dans le source LaTeX, un alinéa terminé par `sur le lapin. se conclura désormais par sur~le~\bname{lapin}.`

Le hic, c'est qu'après avoir écrit les deux regex correspondantes (pour la recherche et pour le remplacement, respectivement), le CR s'appliquera à tous les alinéas, dont ceux d'une à trois lignes. C'est idiot, encore que sans conséquence, pour les alinéas d'une ligne, souvent des répliques de dialogue. Pour les autres alinéas très brefs, c'est en revanche courir le grand risque de donner naissance à d'atroces lignes lavées. Il apparaît donc plus sage de renoncer à empêcher à tout prix les lignes à voleur pour ces alinéas.

Pour cela, il faut enrichir la regex initiale en n'appliquant ce traitement que si la première espace que nous cherchions se trouve précédée d'au moins un certain nombre de caractères, mettons 250 : on exclut ainsi du CR les alinéas trop courts.

8. Eugène BOUTMY, *Dictionnaire de l'argot des typographes*.

Conclusion

J'espère avoir montré l'intérêt des regex pour la préparation de copie en vue d'améliorer la composition typographique. Plus encore que le typographe, le correcteur tirera profit des expressions rationnelles, qui permettent de corriger des centaines d'erreurs courantes. Dans un document non stylé, les regex, combinées aux options de recherche d'un traitement de texte sur la mise en forme, permettront d'identifier avec un faible risque d'erreur les titres auxquels appliquer des styles correspondant à leur niveau hiérarchique.

Apprendre à utiliser les expressions rationnelles n'est pas si difficile pour les usages qui nous concernent. Au reste, ouvrages et ressources en ligne ne manquent pas. On peut même s'y essayer dans son navigateur Web sans risquer de saccager ses documents. Lancez-vous !

Aller plus loin

Vincent FOURMOND, *les Expressions régulières par l'exemple*, Paris, H&K, 2005.

Jeffrey E. F. FRIEDL, *Maîtrise des expressions régulières*, 2^e éd., Paris, O'Reilly, 2003. Épuisé, l'ouvrage est le plus pointu publié en français sur le sujet ; on le trouve d'occasion.

Sites en anglais :

— <https://www.regular-expressions.info/> ;

— <https://www.rexegg.com/>.

Tester ses regex en ligne : <https://regex101.com/>.