Recherche d'imperfections typographiques avec LuaLaTeX

Daniel Flipo
daniel.flipo@free.fr

1 De quoi s'agit-il ?

L'extension `lua-typo` décrite ci-dessous [1] permet de mettre en lumière par un changement de couleur, les lignes potentiellement imparfaites d'un fichier PDF produit par LuaLaTeX ; comme son nom l'indique elle ne fonctionne qu'avec LuaLaTeX. Une liste des pages concernées est affichée à la fin du fichier `.log`, permettant un accès rapide aux pages incriminées. `lua-typo` crée également un fichier de suffixe `.typo` regroupant les informations (type, page, ligne) sur les imperfections relevées.

Normalement, c'est-à-dire lorsque la justification n'est pas trop étroite, (Lua)TeX fait du bon travail, surtout si `microtype` est utilisé mais il peut rester des points à vérifier, notamment des lignes trop pleines ou lavées (*Overfull, Underfull box*), des veuves et des orphelines, des mots coupés en fin de page ou d'alinéa ou sur plusieurs lignes consécutives, des dernières lignes d'alinéa trop courtes ou presque pleines, des pages quasi vides. La répétition d'un même mot ou partie de mot au début ou à la fin de deux lignes consécutives est aussi détectée. La présence en fin de ligne de certains mots très courts (une ou deux lettres, liste dépendant de la langue) peut également être recherchée.

Important : a) les lignes totalement ou partiellement coloriées par `lua-typo` le sont uniquement pour *attirer l'attention* du correcteur à qui il appartient de décider si ces lignes doivent être remaniées ou non. Certains « défauts » peuvent être acceptables dans certaines conditions (multicolonnage, documents techniques) et pas dans d'autres, œuvres litéraires par exemple. Seul un humain entrainé peut décider si une ligne légèrement lavée est acceptable ou non, ou si la suppression d'une coupure malvenue ne va pas provoquer d'autres désordres bien plus graves.
b) Inversement, il n'est pas exclu que `lua-typo` comporte des bogues l'empêchant de détecter des lignes potentiellement imparfaites.

`lua-typo` est largement configurable et devrait pouvoir s'adapter aux exigences variables des auteurs ou correcteurs : voir ci-dessous la liste des options et le fichier de configuration `lua-typo.cfg`.

`lua-typo` ne corrigeant aucun des « défauts » relevés, quels sont les moyens à notre disposition pour le faire manuellement ? Reformuler une phrase est souvent efficace, un auteur peut se le permettre, un correcteur ne le peut pas. Il est possible de jouer sur l'espace inter-mots grâce à la commande TeX `\spaceskip`, ou sur l'interlettrage grâce à la commande `\textls` de `microtype` mais dans les deux cas, il faut le faire *avec parcimonie*[2] pour que le remède ne soit pas pire que le mal. Un *léger* accroissement de l'espace inter-mots ou de l'interlettrage dans un groupe de mots peut rendre acceptable la dernière ligne initialement trop courte d'un paragraphe ou, si elle était presque

1. Version v.0.85, mise à jour le 2023-09-13.
2. Le but ultime est de parvenir à un « gris typographique », aussi parfait que possible ; l'argument optionnel de `\textls` ne devrait guère sortir de l'intervalle $[-5,+15]$ (unité $1/1000^{e}$ `em`) et `\spaceskip` devrait rester très proche de l'espace-mot standard déterminée par les `\fontdimen`.

pleine, ajouter une ligne, ce qui peut permettre de supprimer une orpheline. De même une légère contraction peut supprimer la dernière ligne (courte) d'un paragraphe et éviter une veuve en début de la page suivante.

Je conseille de n'appliquer `lua-typo` que sur des textes « presque au point », d'améliorer ce qui peut l'être puis de *supprimer* l'appel à `lua-typo` afin de ne pas risquer de mettre en lumière les imperfections que l'on aura renoncé à corriger. Pour appliquer toutes les vérifications proposées par `lua-typo`, il suffit d'ajouter dans le préambule la ligne `\usepackage[All]{lua-typo}`

La version courante (0.85) nécessite un noyau LaTeX récent, 2022/06/01 ou ultérieur. Ceux qui ne disposent que d'un noyau antérieur à 2021/06/01 reçoivent un message d'erreur « `Unable to register callback` » ; une version « rollback » est prévue à leur intention, elle se charge par la commande `\usepackage[All]{lua-typo}[=v0.4]`. Une autre version intermédiaire est présente, elle se charge avec l'option `[=v0.65]`.

Les fichiers `demo.tex` et `demo.pdf` fournissent un exemple du traitement opéré par `lua-typo`.

Un grand merci à Jacques André et Thomas Savary pour avoir accepté de tester les pré-versions et pour leurs retours riches et toujours pertinents ; leurs suggestions et leurs encouragements ont grandement contribué à améliorer la première version mise en ligne. Merci également à Michel Bovani pour ses remarques et suggestions qui ont conduit à la version 0.61.

2 Utilisation

Comme indiqué plus haut la vérification la plus complète s'obtient par :
`\usepackage[All]{lua-typo}`

Il est possible de choisir les tests à activer de deux manières, soit « tout sauf … » soit « seulement ceci et cela ». Pour tout activer sauf les options *<OptX>* et *<OptY>* :
`\usepackage[All, <OptX>=false, <OptY>=false]{lua-typo}`
ou pour se limiter aux tests *<OptX>* et *<OptY>* :
`\usepackage[<OptX>, <OptY>]{lua-typo}`

La liste des options et le type des vérifications proposées sont présentés dans le tableau page suivante. Par exemple, pour limiter les vérifications aux lignes trop pleines ou creuses, il suffit de coder :
`\usepackage[OverfullLines, UnderfullLines]{lua-typo}`
Pour tout vérifier sauf les coupures répétées en fin de ligne on codera :
`\usepackage[All, RepeatedHyphens=false]{lua-typo}`
Notez que l'option that `All` doit être la première de la liste, les suivantes étant rétirées de la liste complète définie par `All`.

Le nom des différentes options n'étant pas facile à mémoriser, il est possible de les retrouver sans devoir consulter la documentation ; l'option `ShowOptions` affiche la liste complète dans le fichier `.log` : `\usepackage[ShowOptions]{lua-typo}`

L'option `None`, empêche toute vérification : `\usepackage[None]{lua-typo}` a pour effet de supprimer complètement tout ajout de code LuaTeX (aucune fonction n'est ajoutée aux *callbacks* de LuaTeX). Cette option peut-être utile lors de la toute dernière compilation, elle n'est pas tout-à-fait équivalente à la mise en commentaire de la ligne car les variables utilisées par `luatypo` restent définies ; si certaines ont été modifiées

Nom	Imperfection à signaler
`All`	Active toutes les options ci-dessous
`ShortLines`	Dernière ligne d'alinéa trop courte ?
`BackParindent`	Dernière ligne d'alinéa *presque* pleine ?
`ShortPages`	Page quasi vide (quelques lines) ?
`OverfullLines`	Ligne trop pleine ?
`UnderfullLines`	Ligne lavée ?
`Widows`	Veuve (haut de page) ?
`Orphans`	Orpheline (bas de page) ?
`EOPHyphens`	Mot coupé en bas de page ?
`RepeatedHyphens`	Coupures sur trop de lignes consécutives ?
`ParLastHyphen`	Coupure à l'avant-dernière ligne d'un alinéa ?
`EOLShortWords`	Mots courts (1 or 2 lettres) en fin de ligne ?
`FirstWordMatch`	Même (partie de) mot en début de lignes consécutives ?
`LastWordMatch`	Même (partie de) mot en fin de lignes consécutives ?
`FootnoteSplit`	Fin de note de bas de page sur page suivante ?
`ShortFinalWord`	Mot de fin de phrase court en haut de page
`MarginparPos`	Note marginale se terminant trop bas

dans le préambule aucun message d'erreur du type "*Undefined Control Sequence*" ne sera émis à leur sujet.

Terminons par quelques précisions sur ces options.

`FirstWordMatch :` les répétitions en début de ligne dans les listes ne sont pas signalées. Ceci est voulu car elles résultent d'un choix délibéré de l'auteur.

`ShortPages :` lorsque le nombre de lignes d'une page est jugé insuffisant (voir ci-dessous), seule la dernière ligne de celle-ci est mise en couleur.

`RepeatedHyphens :` de même, lorsque le nombre de lignes consécutives affectées par des coupures dépasse le seuil fixé (voir ci-dessous), ne sont coloriées que les coupures en excès.

`ShortFinalWord :` lorsque le premier mot de la première ligne d'une page termine une phrase et qu'il est court (au plus `\luatypoMinLen=4` lettres), on le signale.

3 Paramétrage personnalisé

Pour certaines vérifications faites par `lua-typo` un paramétrage est nécessaire : à partir de quelle limite une dernière ligne d'alinéa est-elle considérée comme trop courte ? Combien de coupures consécutives en bout de ligne sont-elles acceptables ? Ces réglages dépendent évidemment du contexte, un correcteur de romans aura des exigences plus strictes qu'un auteur de documentation technique par exemple…

`lua-typo` permet de modifier le réglage des curseurs soit dans le fichier `lua-typo.cfg` soit dans le préambule après l'appel de `lua-typo` ; les réglages placés dans le préambule prévalent sur ceux du fichier `lua-typo.cfg` qui eux-mêmes prévalent sur les réglages internes de l'extension.

Le fichier `lua-typo.cfg` fourni avec la distribution reprend exactement les réglages internes, il se trouve normalement dans le répertoire TEXMFDIST des distributions

TeXLive, MikTeX, etc. L'utilisateur a la possibilité de recopier ce fichier soit dans son répertoire de travail, soit dans son répertoire TEXMFHOME ou TEXMFLOCAL et de le personnaliser comme il l'entend.

Voici la liste complète des paramètres personnalisables avec leur valeur par défaut, leurs noms sont systématiquement préfixés par `luatypo` afin d'éviter de possibles conflits avec d'autres extensions.

`BackParindent :` la dernière ligne d'un alinéa devrait, soit se terminer à plus de `\luatypoBackPI=1em` de la marge droite, soit être (approximativement) pleine (tolérance `\luatypoBackFuzz=2pt`)[3].

`ShortLines : \luatypoLLminWD=2\parindent`[4] fixe la longueur minimale acceptable pour la dernière ligne d'un alinéa.

`ShortPages : \luatypoPageMin=5` fixe le nombre minimal de lignes d'une page pour que celle-ci ne soit pas déclarée trop courte. En fait, la position de la dernière ligne est prise en compte afin que les pages de titre ou celles contenant une image ne soient pas signalées comme fautives.

`RepeatedHyphens : \luatypoHyphMax=2` fixe le nombre maximal acceptable de lignes consécutives terminées par un mot coupé.

`UnderfullLines : \luatypoStretchMax=200` fixe le pourcentage maximal acceptable pour l'étirement des espaces-mots, au-delà la ligne est déclarée lavée. La valeur donnée doit être un entier supérieur ou égal à 100, cette valeur 100 correspond à l'étirement maximal prévu par la fonte (`\fontdimen3`); avec ce réglage attendez-vous à trouver une kyrielle de lignes creuses! En fait la valeur par défaut (200) correspond approximativement à ce que TeX, avec les réglages par défaut (`\tolerance=200`, `\hbadness=1000`), considère comme *Underfull hbox*.

`First/LastWordMatch: \luatypoMinFull=3` et `\luatypoMinPart=4` nombres minimaux de lettres identiques (resp. pour un mot complet ou pour une partie de mot) au début ou à la fin de deux lignes consécutives déclenchant l'avertissement. Avec ce réglage (3 et 4), seront détectées deux lignes se terminant par « cible » et « irrésistible » ou « irrésistible-(ment) » (quatre lettres en commun), ainsi que la présence de « mon » en début ou fin de deux lignes consécutives (trois lettres en commun), mais « mon » et « mont » en début de ligne échappent à la détection.

`EOLShortWords:` cette option signale la présence en fin de ligne de mots très courts (une ou deux lettres) qui sont répertoriés dans une des listes suivantes (elles dépendent de la langue courante) :

```
\luatypoOneChar{<langue>}{'<liste de mots>'}
\luatypoTwoChars{<langue>}{'<liste de mots>'}
```

Lorsque les listes correspondant à la langue du document sont vides, aucune vérification n'est effectuée. Pour l'instant, il y a deux lignes (non actives) prévues pour le français :

```
\luatypoOneChar{french}{'À Ô Y'}
\luatypoTwoChars{french}{'Je Tu Il On Au De'}
```

Deux contraintes sont à respecter lorsqu'on veut personnaliser ces listes :
a) le premier argument (langue) *doit être connu de* `babel`, aussi les commandes

3. Certains auteurs n'acceptent pas les lignes pleines en fin de paragraphe, ceux-là pourront faire `\luatypoBackFuzz=0pt` pour qu'elles soient détectées comme fautives.
4. Ou `20pt` si `\parindent=0pt`.

`\luatypoOneChar` et `\luatypoTwoChars`, si elles sont utilisées, doivent l'être *après* le chargement de `babel`, une bonne habitude à prendre est donc de toujours charger `lua-typo` *après* `babel` ; b) le second argument *doit être une chaîne de caractères*, donc entourée de simples ou doubles *quotes* ASCII et composées de mots séparés par des espaces comme dans les exemples ci-dessus.

`\luatypoMarginparTol` est une *dimension* qui vaut `\baselineskip` par défaut ; les notes marginales qui se terminent à plus de `\luatypoMarginparTol` en dessous de la dernière ligne de la page sont déclarées fautives.

À chacune des vérifications faites par `lua-typo` peut être attachée une couleur spécifique pour mettre en évidence les imperfections détectées. Actuellement, seulement six couleurs sont utilisées par défaut, voici leur définition dans `lua-typo.cfg` :

```
% \definecolor{LTgrey}{gray}{0.6}
% \definecolor{LTred}{rgb}{1,0.55,0}
% \definecolor{LTline}{rgb}{0.7,0,0.3}
% \luatypoSetColor1{red}      % Coupure à l'avant-dernière ligne
% \luatypoSetColor2{red}      % Coupure en bas de page
% \luatypoSetColor3{red}      % Coupures consécutives
% \luatypoSetColor4{red}      % Mot court en fin de ligne
% \luatypoSetColor5{cyan}     % Veuve
% \luatypoSetColor6{cyan}     % Orpheline
% \luatypoSetColor7{cyan}     % Dernière ligne d'alinéa trop courte
% \luatypoSetColor8{blue}     % Ligne trop pleine
% \luatypoSetColor9{blue}     % Ligne creuse
% \luatypoSetColor{10}{red}   % Page presque vide (qq. lignes)
% \luatypoSetColor{11}{LTred} % Répétitions en début de ligne
% \luatypoSetColor{12}{LTred} % Répétitions en fin de ligne
% \luatypoSetColor{13}{LTgrey}% Dernière ligne alinéa presque pleine
% \luatypoSetColor{14}{cyan}  % Note de bas de page éclatée
% \luatypoSetColor{15}{red}   % Mot de fin de phrase en haut de page
% \luatypoSetColor{16}{LTline}% Ligne présentant plusieurs "défauts"
% \luatypoSetColor{17}{red}   % Note marginale se terminant trop bas
%
```

`lua-typo` charge les extensions `luacolor` et donc `color`. Seules les couleurs portant un nom (*named colors*) peuvent être utilisées dans la commande `\luatypoSetColor` ; pour en définir de nouvelles il faut donc, soit utiliser la commande `\definecolor` de l'extension `color` (comme ci-dessus pour `LTgrey` ou `LTred`), soit charger l'extension `xcolor` qui donne accès à une kyrielle de noms de couleurs.