



---

IFT6285 (TALN) — Devoir2  
Correction de mots

---

Contact :  
**Philippe Langlais** +1 514 343 61 11 ext: 47494  
RALI/DIRO [felipe@iro.umontreal.ca](mailto:felipe@iro.umontreal.ca)  
Université de Montréal <http://www.iro.umontreal.ca/~felipe/>

■ dernière compilation : 17 octobre 2023 (11:56)

## Contexte

---

Jusqu'à récemment, la correction orthographique était un domaine de recherche plutôt développé dans un contexte industriel. Dans ce devoir nous allons aborder la correction de mots comme un problème d'identification des voisins d'un mot erroné, le candidat à la correction étant sélectionné dans ce voisinage. Une excellente source d'inspiration est le [blog](#) de Peter Norvig sur le sujet. Vous pouvez si vous le souhaitez reprendre tout ou partie de son [code](#), ou l'implémentation [pyspellchecker](#).

## Données

---



Vous avez accès à un texte [typo-0.2.txt](#) dans lequel des typos ont été introduites. Ce texte est déjà découpé en mots (l'espace est le séparateur) et les typos sont marquées par des balises `<typo>` et `</typo>` avec un attribut `orig` indiquant le mot original. Votre but est de tenter de retrouver la forme originale de chaque typo marquée, à l'aide des distances vues en cours ou variantes (vous ne devez bien sûr pas utiliser l'attribut `orig`).

Vous disposez également du lexique [voc-1bwc.txt](#) des 201 315 mots rencontrés au moins 16 fois dans la partie d'entraînement du corpus 1BWC (CHELBA et al. 2013), ayant une longueur comprise entre 3 et 29 caractères, ne contenant aucun chiffre, contenant au moins une lettre et ne terminant pas par un point.<sup>1</sup> Chaque mot est accompagné de sa fréquence dans 1BWC (un espace sépare la fréquence du mot).

## À faire

---

1. Vous devez écrire un programme qui lit un lexique (à priori `voc-1bwc.txt`) et un fichier texte au même format que décrit et qui produit pour chaque typo marquée les corrections **les plus plausibles** au format prescrit par l'exemple qui suit (seule la sortie nous intéresse ici). Notez que

---

1. Plus précisément, la commande suivante a été lancée : `cat $in/news* | tr ' ' '\n' | tr '[:upper:]' '[:lower:]' | sort | uniq -c | sort -k1,1n | awk '$1 >15 && length($2) > 2 && length($2) < 30 && $2 ! /[[:digit:]]/ && $2 ~ /[a-z]/ {print $0}' | grep -v '\-\' | grep -v '\.$'` où `$in` vaut le répertoire de 1BWC installé au DIRO (voir devoir 1).

les corrections sont affichées en ordre décroissant de plausibilité (la correction la plus pertinente en premier) et que vous pouvez indiquer au plus 5 corrections.

```
head -n 10 typo-0.2.txt | \  
corrige -lexicon=voc-1bwc.txt > typo-10.sol
```

Aucune liberté n'est donnée quant aux espaces dans votre sortie : chaque phrase en entrée produit une phrase en sortie, sur une seule ligne où chaque mot est séparé d'un autre par exactement un espace, les corrections dans les éléments **corrections** sont séparées par un espace. Le début de la première ligne de la sortie :

```
Q. Doesn 't this just hit the very <correction  
orig="wealthy" typo="wealtohy">wlad weld wltw  
wallowed walt</correction> who can
```

indique que la typo **wealtohy** a été introduite en remplacement du mot **wealthy** après la chaîne **Q. Doesn 't this just hit the very** et que cette typo a été (à tort) reconnue comme **wlad**, ou comme **weld**, etc. dans cet ordre. La correction produite commencerait donc par **Q. Doesn 't this just hit the very wlad who can**


L'approche demandée dans ce devoir consiste à chercher pour un mot à corriger un ensemble de voisins au sens d'une distance particulière (vous en considérerez au moins trois vues en cours) et à classer ces corrections soit directement à l'aide de la ou des distances, soit à l'aide du modèle **unigramme** que constitue le lexique passé à votre programme, soit avec les deux. Votre but est d'étudier l'impact des distances utilisées pour calculer les voisins et de décider d'une meilleure approche au problème de correction dans ce contexte. Votre meilleure approche peut combiner plusieurs distances et peut également faire usage de programmes existants (ce qui n'empêche pas que vous devez tester au moins 3 distances vues en cours).

2. Vous devez écrire un programme d'évaluation qui prend une sortie produite par votre programme de correction (donc au format prescrit) et qui produit une ou plusieurs mesures de la qualité de votre correcteur.
3. Au plus tard dimanche 22 octobre, un nouveau fichier avec des typos vous sera distribué. Vous devrez corriger ce fichier à l'aide de

votre programme puis soumettre votre correction. Attention, la sortie générée par votre programme doit être compatible avec la sortie prescrite, car c'est moi qui vais évaluer vos sorties. Si vous avez un doute sur le format, testez votre sortie avec le programme `eval.py` par une commande comme :

```
cat <votre-sortie> | python3 eval.py
```

où `<votre-sortie>` est le fichier dont vous souhaitez vérifier le format (dans l'exemple, `typo-0.12.sol`). Si le programme s'arrête sans générer de message d'erreur (ou sans s'interrompre), alors votre format est à priori compatible. Une option `--verbosity` permet d'afficher les lignes traitées.

4. Produire un rapport (format pdf, en français ou en anglais) **d'au plus 3 pages** qui contient les informations suivantes :
  - (a) les codes/librairies que vous utilisez le cas échéant, 
  - (b) un descriptif concis et précis des métriques calculées dans `eval.py` et la ou les métriques que vous avez tenté d'optimiser en développant votre correcteur.
  - (c) une courte description des différentes distances étudiées. Je ne vous demande pas de m'expliquer le détail des distances, votre description doit être concise et pointer au besoin sur une ressource Web les décrivant (si disponible bien sûr). Si vous utilisez une implémentation existante, indiquez-le.
  - (d) une description de votre correcteur indiquant comment les corrections sont sélectionnées et classées.
  - (e) l'impact des distances étudiées sur les temps de correction, sur les performances.
  - (f) une description de votre meilleure approche et de ses limites.

Le nom du fichier pdf contenant votre rapport doit contenir les noms des personnes y ayant contribué (ex : `rapport-<nom1>-<nom2>.pdf`) et ces mêmes noms doivent figurer au début du rapport.

## Remise

---

La remise est à faire sur Studium sous le libellé `devoir2`. Vous devez remettre votre code, votre rapport (format pdf, texte en anglais ou en français) et la sortie de votre programme de correction dans une archive (gzip, tar, ou tar.gz) dont le nom est préfixé de `devoir2-name1` ou `devoir2-name1-name2` selon que vous remettez seul ou a deux, où `name1` et `name2` sont à remplacer par l'identité des personnes faisant la remise (`prénom.nom`). Aussi bien votre rapport que votre sortie doivent porter le nom des personnes concernées par la remise. Donc si j'avais à remettre seul ma solution au devoir2, je le ferais sous le nom `devoir2-philippe_langlais.tar.gz` avec dedans mon rapport `rapport-philippe_langlais.pdf`, la sortie de mon programme (ex : `sortie-philippe_langlais.txt`) et mes codes. Le devoir est à remettre en groupe d'au plus deux personnes au plus tard mercredi 25 octobre à 23h59.

---

CHELBA, Ciprian et al. (2013). “One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling”. In : *CoRR* abs/1312.3005. URL : <http://arxiv.org/abs/1312.3005>.