Correcteur orthographique

Maximilien Rigaut

December 12, 2012

Abstract

Le programme présenté est un correcteur orthographique à but purement pédagogique et démonstratif. Cependant, une attention a été donnée quand à la possibilité d'adapter ce code a divers projets, et applications pratiques. Certaines impasses furent faites au profit de la vitesse, là ou l'on eu pût espérer une plus grande rigueur.

Le code produit fut documenté au cours de sa rédaction, le logiciel doxygen, permet de l'extraire et en former une documentation au format html et latex. Celle-ci est disponible dans les dossier doc/html et doc/latex, et peut parfois apporter des éclaircissement sur les méthodes utilisées.

Contents

1 Installation **2**

2 Résultats **2**
- 2.1 Tests . . . 2
- 2.2 Résultats de l'exécution . . . 2
- 2.3 Explications des Résultats . . . 3
  - 2.3.1 Contexte . . . 3
  - 2.3.2 Homophonie . . . 3
  - 2.3.3 Cas particulier . . . 3

3 Utilisations **4**
- 3.1 Fonctionnement . . . 4
- 3.2 Configurations alternatives . . . 4

4 Choix de mise en œuvre **4**
- 4.1 Distance de Levenshtein . . . 4
- 4.2 Hachage . . . 4

5 Améliorations **4**
- 5.1 Efficience . . . 4
- 5.2 Vitesse . . . 4

1 Installation

Pour configurer programme se référer à la section 3.2 page 4. Les commandes nécessaires a la compilation sont:

```
make init
make cleaner all
make doc
./bin/corrector ressources/dico.txt ressources/fautes2012.txt
```

Nb: Ne pas s'inquiéter des terribles warnings de doxygen lors de la création de la documentation. La compilation et l'exécution furent été testée sous Linux(Lubuntu) et MacOS(Snow leopard), si le programme ne parvenait pas a compiler ou s'éxécuter correctement, vous pouvez le notifier à max.rigaut[AT]orange.fr ou sur https://github.com/Malphaet/Maximilien-Rigaut/issues

2 Résultats

Les résultats sont donnés pour le fichiers de fautes fourni, celui-ci a été légèrement modifié. Tout d'abord la syntaxe a été changée en faveur d'un '¿'. De plus certains mots furent écartés de la liste, ou corrigés, ceux-ci étant (à mon souvenir) 'pénate', ainsi qu'un mot possédant un espace avant le retour de ligne. En effet, quelque soit la nature de la faute à corriger, aucun correcteur ne peut proposer un mot qui ne serait présent dans son dictionnaire.

2.1 Tests

Tests préliminaires (Voir section 3.2 page 4 pour ne pas afficher ces tests). Ces derniers visent a donner un aperçu des capacités du logiciel et ne doivent donc pas être considérés comme partie intégrante du programme.

```
> Writing test results at ./tmp/tests.txt
> Testing hash function
Time elapsed: 593 ms. Hash speed: 168634 hashs per ms
> Testing hash dict building function
Digesting dictionnary: ./ressources/dico.txt
Time elapsed: 190 ms, alpha: 0.404910, word digested: 424579, worst collision: 5
Hash repartition :
k Number    Fisher
0 349251 283209.27
1  65718 114674.83
2   8715  23216.61
3    833   3133.56
4     57    317.20
5      5     25.69
> Testing 3-tuple reduction function
Digesting dictionnary: ./ressources/dico.txt
Time elapsed: 643 ms, alpha: 4.340960, tuples digested: 4551826, worst collision: 70559
```

2.2 Résultats de l'exécution

Avec un nombre de suggestion de 10, les résultats sont les suivants:

```
Starting analysis...
Analysing aigü...(aigu) led to no match

Analysing ayant-droits...(ayants droit) led to no match
avant-toits avant-projets avant-gouts avant-goûts avant-monts avant-ports avant-toit
Analysing capoiera...(capoeira) led to no match
capotera caponnera carroiera chatoiera capoterai capoteras captivera clapotera crapotera cabote
Analysing connection...(connexion) led to no match
connections confection convection confections connectiez connectif connectifs connective convec
Analysing Lybie...(Libye) led to no match
Lycie Lydie Lydia Lyme Lucie
Analysing oyé...(oyez) led to no match
ohé olé osé ové oxydé ocré ollé ondé opté orné
Analysing penate...(pénates) led to no match
pente pendante pensante prenante penalty penaude phénate pendage pennage pentane
Analysing puis ensuite...(puis) led to no match

Analysing raisonnance...(résonance) led to no match
raisonnante raisonnantes raisonnant raisonnants raisonnable raisonnasse raisonnables raisonnass
Analysis complete, lasted 4796ms.
196 words were first guess, 62 were amongs the guesses and 9 weren't found.
     M      F      N
73.41% 23.22%  3.37%
```

2.3 Explications des Résultats

2.3.1 Contexte

Certaines erreurs ne peuvent être corrigées qu'en fonction du contexte, j'entends par la principalement le mot 'raisonnance'. Quand on analyse les résultats:

```
raisonnante raisonnantes raisonnant raisonnants raisonnable raisonnasse raisonnables raisonnass
```

On remarque que ceux-ci sont bel et bien proche du mot donné, et seul un contexte permet de réellement départager.

2.3.2 Homophonie

Dans plusieurs cas, seule une analyse phonétique pourrait permettre de rajouter des mots plus pertinents a la liste des suggestions. En effet, connection (connexion), raisonnance (résonance), oyé (oyez), aigü (aigu) ou alors Lybie (Libye) sont des homophonies. Bien souvent les suggestion, bien que très pertinentes, ne tiennent pas compte de ce fait.

2.3.3 Cas particulier

Les mots 'injustement' composés ne peuvent être traités par cette mouture du correcteur, puisque nécessitant l'usage de deux mots.

3 Utilisations

3.1 Fonctionnement

S

3.2 Configurations alternatives

La plupart des configuration importantes relatives au programme sont situés dans inc/corrector.h, certaines fonctionnalités plus poussées peuvent être éditées dans inc/wordtools.h

4 Choix de mise en œuvre

4.1 Distance de Levenshtein

La fonction implémentée ce distingue de la fonction originelle en cela quelle renvoie directement le pourcentage d'éloignement quand au mot originel (plutôt qu'une simple distance).

4.2 Hachage

La fonction jhash, variante de la fonction de hachage de java, permet de hacher sur un nombre donné de bits d'entropie. Celle-ci ce distingue par son choix quand a la méthode pour conserver uniquement ce degré d'entropie.

Lorsque l'algorithme choisi par java choisi d'appliquer un modulo a chaque étape, on utiliser a la place, choisir d'effectuer une multiplication sur la lettre concernée, puis de découper le résultat en 'morceaux' de la taille désirée en enfin d'appliquer un ou exclusif entre eux.

On remarque que cette fonction donne environ $\pm 70\%$ de répartition des hashs contre, $\pm 150\%$ pour le modulo. Le but étant une variation de 0% (un hash parfaitement uniforme).

Cette implémentation est en moyenne 23% plus rapide que l'implémentation avec modulo, et produit entre +2% et -120% de collisions en moins par rapport au modulo simple.

4.3 Correction

La correction ce fait par étapes successives, tout d'abord les triples présents dans le mot donnent naissance a une liste de mots a corriger. Durant cette étape, si il advenait qu'un des caractères spéciaux soit rencontré (ici 'oe'), alors, le caractère est ajouté a la liste des possibilités. Ensuite,si leur nombre de triplets communs est suffisamment grand, les mots ajoutés seront classés selon leur distance de Levenshtein au mot donné. Il est a noter que la distance au mot donné, puisque effectuée en sur un grande liste, l'utilisation des threads a été retenue, permettant d'éviter la formation d'un goulot d'étranglement en cas de trop grand nombre suggestions.

5 Améliorations

5.1 Efficience

La plupart des mots non corrigés sont des pistes d'amélioration, ainsi la présence d'un 'ct' dans un triple pourrait nécessiter l'ajout d'un 'x' a la dans la liste des suggestions. Si le mot suggéré est vraiment trop éloigné Levenshtein ce chargera de l'éliminer des candidats.

5.2 Vitesse

Une autre piste d'amélioration est directement lié a l'usage final du programme. En effet, si celui-ci ce doit d'être interfacé avec l'homme, alors a chaque interaction avec ce dernier, un fil d'exécution parallèle (thread) peut prendre le relaie et corriger le reste des suggestions en attendant l'utilisateur.

Si toutefois le dictionnaire devait ce destiner a scanner des textes pour y déceler des fautes et proposer des suggestions, alors il faudrait plutôt mettre en place un ensemble de threads œuvrant a ce partager la charge de travail. La charge processeur moyenne lors des exécutions est de 40% autorisant une performance bien accrue lors d'un multithreading intensif.