POLYTECH NICE-SOPHIA

SI 3 - 2013-2014

Erick Gallesio

Langage C

Feuille 6
Correcteur orthographique

1 Introduction

Dans cette feuille nous allons construire un correcteur orthographique. Notre correcteur sera capable de corriger des mots comportant une seule erreur (un caractère oublié, un caractère en trop, une inversion de lettre, ...).

Avant de commencer, voyons comment le programme est organisé. Tout d'abord, pour éviter de gérer les problèmes d'accents et d'encodage qui en découlent, nous nous limiterons à écrire un correcteur pour l'anglais. Le principe de fonctionnement est assez simple: étant donné un mot, nous essayons de choisir la correction orthographique la plus probable pour ce mot (la "correction" peut-être le mot d'origine). Il n'existe aucun moyen de savoir avec certitude le mot le plus probable (par exemple, si on a le mot "larges", celui-ci doit il être remplacé par "larger" ou "largest"?). Par conséquent nous allons utiliser un mécanisme probabiliste pour déterminer le mot de remplacement qui semble le plus judicieux. Nous allons donc construire une liste de toutes les corrections possibles pour ce mot et choisir parmi celles-ci la correction qui a la plus forte probabilité.

Considérons le mot mal orthographié m = "*thei*", et les deux corrections possibles c_1 = "*the*" (l'utilisateur a tapé une lettre en trop) et c_2 = "*they*" (l'utilisateur a tapé une lettre à la place d'une autre). Ici, notre programme choisira de remplacer m par c_1, car le mot *"the"* est plus employé en anglais que *"they"*.

2 Principes

2.1 Corpus

La première tâche consiste à se construire une table contenant des mots (anglais ici) et à leur associer un poids en fonction de leur fréquence dans la langue. Il existe de telles tables toutes faites, mais nous allons ici la créer nous même à partir d'un texte d'un volume conséquent. Pour cela, nous allons prendre l'Illiade d'Homère et construire une table qui associera pour chaque mot son nombre d'apparitions dans le texte. Cela constituera notre corpus sur lequel nous nous baserons pour nos corrections orthographiques.

Pour la réalisation de cette partie nous utiliserons les primitives Posix `hcreate`, `hsearch` et `hdestroy` qui permettent de gérer des tables de *hachage* (tables dont l'accès est généralement très efficace). Pour comprendre comment utiliser ces tables, il vous suffit de regarder la page de manuel de `hcreate` qui contient un exemple complet.

Note: Pour faciliter les comparaisons, tous les mots qui seront rangés dans la table devront être convertis en minuscules auparavant.

2.2 Corrections

Voyons maintenant comment énumérer l'ensemble des corrections possibles pour un mot m. Nous allons considérer ici des mots qui ne contiennent qu'une erreur. Une correction peut être:

- un *effacement* (suppresion d'une lettre)
- une *transposition* (échange de deux lettres adjacentes)
- une *modification* (changement d'une lettre du mot m par une autre lettre de l'alphabet)
- une *insertion* (insertion d'une lettre de l'alphabet dans toutes les postions possibles de m.

Le nombre de corrections peut être assez conséquent puisque pour un mot de longueur n, nous avons n effacements, $n-1$ transpositions, $26n$ modifications et $26(n+1)$ insertions. Par conséquent, le nombre de corrections potentielles pour un mot de longueur n est $54n + 25$ (soit tout de même 565 corrections pour un mots de 10 lettres!).

2.3 Correction d'un mot

Pour corriger un mot m nous allons utiliser un algorithme simple:

- si m est dans notre corpus, on renvoie ce mot.
- si le m n'est pas présent,
  - on construit l'ensemble E des corrections de m tel qu'il est défini dans la section précédente.
  - on revoie le mot de E qui est le plus probable (i.e. celui qui a le plus grand nombre d'apparitions).
  - si aucune correction de m n'apparaît dans notre corpus, on renvoie m (qui après tout n'est peut-être pas faux).

3 Travail demandé

Prendre le fichier de ressources associé à ce TD, il contient le programme principal `main.c`, ainsi que les fichiers d'entêtes `corpus.h`, `correct.h`, `hash.h` et une partie de `correct.c`. Le fichier `correct.h` contient des exemples exhaustifs sur les corrections à construire pour un mot de 4 lettres. Par ailleurs, il vous est aussi donné deux fichiers de tests: `test1.c` et `test2.c`. Le premier permet de tester la table de *hash* et le second permet de tester la création du corpus en partant du fichier `Data/strtok.txt`. Pour votre programme final le corpus sera construit à partir de `Data/Iliad.txt`
Vous devez travailler en binômes.
Ce que vous devez rendre: un programme complet (avec son Makefile) et un README indiquant ce qui marche ou pas dans votre programme. N'oubliez pas d'indiquer les membre du binôme. La clarté du code et la qualité de la distribution (makefile correct et complet, respect des conventions issues des fichier ".h", ...) et la correction du programme interviendront dans la notation.
Pour info, voila la taille des fichiers de la correction:

```
$ wc -l corpus.c correct.c hash.c main.c
  43 corpus.c
 173 correct.c
  92 hash.c
  37 main.c
 345 total
```

Date de remise du TD: 22 janvier 2014 – 23h59.