Modélisations Mathématiques Partie 2

William Mocaër, Yanis Ouakrim, Simon Wellenreiter
Info 2 – Groupe 2

2016
IUT de Nantes

Table des matières

1 Introduction **1**

2 Identification de langue **1**
2.1 Spécification 1
2.2 Utilisation 1
2.3 Méthode 2
2.3.1 Implémentation 2

3 Identification du type de texte **3**
3.1 Spécification 3
3.2 Utilisation 3

4 Résultats **4**
4.0.1 Résultat de `quelleLangue` : 4
4.0.2 Résultat de `quelTypeTexte` : 5

5 Pour aller plus loin... **5**

6 Conclusion **5**

1 Introduction

Dans le cadre de la seconde partie du module de *Modélisations Mathématiques*, nous avions à développer un ensemble de scripts en `bash` permettant de traiter de façon automatisée des fichiers textes. Le cours s'articule en deux parties : la reconnaissance linguistique et la reconnaissance "typologique" de textes. L'enjeu principal était de proposer une solution permettant l'identification de la langue d'un fichier texte et la reconnaissance de son type (romanesque, économique, philosophique...).

2 Identification de langue

2.1 Spécification

Nous voulions que notre programme prenne un fichier texte en entrée, par exemple un extrait de roman, et nous donne la langue la plus probable dans laquelle ce texte est écrit.

2.2 Utilisation

Quel que soit le texte utilisé il faut que son nom soit sans espaces, caractères spéciaux ou symboles. Voici les modalités d'utilisation du script d'identification de langue : Le script à exécuter est le script *quelleLangue.bash*, auquel on passe en paramètre le fichier texte. On peut également le passer par un pipe, exemples :

```
./quelleLangue.bash FichierTexte.txt [-all]
```

Ou encore :

```
cat FichierTexte.txt | ./quelleLangue.bash
```

On peut donc tester avec une phrase de la manière suivante :

```
echo "Bonjour, cette phrase est en français, cela marche mieux si les phrases
sont longues" | ./quelleLangue.bash
```

On peut rajouter l'option "-all" (uniquement lorsqu'on passe en paramètre le fichier) qui permet d'afficher les fréquences de toutes les langues, elles correspondent au nombre de mots présents dans la liste des 4000 mots de chaque langue, parmi les 100 mots les plus présents du texte.

2.3 Méthode

Notre méthode est la suivante : nous nous servons des listes des 4000 mots les plus fréquents dans chaque langue fournie. En comparant ces listes avec la liste de mots d'un texte (les 100 mots les plus présents) nous obtenons une valeur égale au nombre de mots présents dans les deux listes, ainsi on obtient une fréquence pour chaque langue ou, plus précisément, une valeur pour chaque liste de 4000 mots d'une langue. On considère que la langue, dont la liste a obtenu le plus grand nombre de mots, est la langue du texte.

2.3.1 Implémentation

Nous avons choisi de décomposer le processus d'identification de la langue d'un texte en deux parties : la première partie est un script (`probaLangue.sh`), qui permet de calculer l'intersection entre :

— les mots les plus utilisés du texte donné en paramètre
— et les mots les plus utilisés d'une langue donnée en paramètre

La deuxième partie est un autre script (`quelleLangue.bash`) qui détermine la langue d'un texte donné en paramètre. Nous avons choisi de découper notre code en plusieurs sous-programmes pour plus de lisibilité. Le code de `quelleLangue.bash` est commenté ligne par ligne.

2.3.1.1 Calcul de l'intersection des mots les plus utilisés d'un texte avec ceux d'une langue

Le script que nous avons créé prend deux arguments, un premier correspondant au code ISO 639-1 d'une langue et un deuxième qui correspond à un fichier texte au format UTF-8. Le script renvoie le nombre de mots du fichier texte qui se trouve dans les 4000 mots les plus utilisés de la langue donnée en paramètre. Afin d'obtenir un tel résultat, nous avons eu recours à la ligne de commande `fgrep`.

2.3.1.2 Détermination de la probabilité de langue la plus élevée

Nous avons ensuite créé un algorithme permettant de trouver, la langue d'un texte par comparaison des intersections entre les 100 mots les plus fréquents d'un texte (en retirant préalablement les caractères spéciaux et les mots non significatifs ou "stop-words") et les 4000 mots les plus fréquents d'une langue.

2.3.1.3 Préparation au traitement

Afin d'obtenir un résultat significatif, il est impératif d'effectuer un premier traitement des fichiers d'entrée avant qu'ils ne soient traités par l'algorithme de reconnaissance linguistique. Dans cette optique, nous faisons subir aux textes d'entrée plusieurs transformations :

— **Conversion en UTF-8**
  — les listes de mots que nous utilisons en tant que corpus d'apprentissage sont des fichiers texte encodés en `UTF-8`. Afin d'être en mesure d'effectuer des comparaisons entre ces deux fichiers, il est essentiel qu'ils aient le même encodage.

— **Mise en minuscule** :
  — tous les mots doivent avoir la même casse de façon à pouvoir être comparés efficacement par l'algorithme.
— **Ajout d'espaces** :
  — Pour éviter que des fragments de mots puissent être assimilés à des mots, nous avons choisi d'encadrer chaque mot par deux espaces. Par exemple, sans espaces le mot *information* serait compté comme une occurrence du mot *formation*.
— **Saut de lignes** :
  — l'ajout de sauts de ligne permet de faciliter l'usage des fonctions de comparaison
— **Suppression des caractères non alphabétiques :**
  — nous cherchons à isoler les mots et les signes de ponctuation auxquels ils pourraient éventuellement être accolés.
— **Suppression des mots de moins de deux caractères** :
  — nous avons choisi de considérer les mots de deux lettres ou moins comme étant non significatifs, on parle de *stopwords*
— **Groupement des occurrences de chaque mot et décompte de leur fréquence**
— **Tri des mots par nombre d'occurrences**
— **Extraction des 100 mots les plus fréquents**

2.3.1.4 Traitement

Une fois le prétraitement effectué on parcourt la liste de langues et pour chaque langue on utilise le programme `probaLangue` pour récupérer le nombre d'occurrences du mot donné en paramètre avec la liste des 4000 mots de chaque langue. On notera que les langues sont classées par nombre d'occurrences et de manière décroissante.

3 Identification du type de texte

3.1 Spécification

Notre programme permet d'obtenir le type auquel semble appartenir un texte. Cependant cela n'est possible que pour un texte français (pour les types déjà enregistrés), il faut donc utiliser au préalable le script précédent pour déduire la langue du texte. Le programme `quelTypeTexte.sh` a pour objectif de trouver le type auquel le texte appartient, c'est à dire :
— Philosophique
— Économique
— Roman
— ...
Il prend en paramètre un fichier texte composé d'extraits d'autres textes. On peut y ajouter l'option `-save [nomDuNouveauTypeTexte]` qui permet de récupérer une sauvegarde du traitement composé des verbes les plus fréquents du texte passé en entrée. Et ainsi chaque nouveau texte sera comparé au nouveau corpus de TypeTexte. Nous verrons plus tard le script `appendreTypeTexte.sh` qui fait la même chose mise à part que nous utilisons tout un dossier de textes et non plus un seul texte.
Il faut cependant noter que certains mots présents dans les 20 verbes les plus fréquents d'un type sont parfois des verbes qui sont utilisés en tant que mot "non-verbe" : par exemple le mot "chose" se retrouve dans les 20 verbes les plus présents en philosophie, s'il se retrouve dans cette liste ce n'est pas parce qu'il est utilisé en tant que verbe (verbe "choser") mais en tant que nom (une chose).

3.2 Utilisation

Quelque soit le texte utilisé il faut que son nom soit sans espaces, caractère spéciaux ou symboles. Le script à exécuter est `quelTypeTexte.sh` avec en paramètre le fichier texte, exemples d'utilisations :

```
./quelTypeTexte.sh FichierTexte.txt [-all]
```

ou bien :

```
cat FichierTexte.txt | ./quelTypeTexte.sh
```

On peut donc tester avec une phrase de la manière suivante :

```
echo "les choses entrent en forme lorsque qu'elles peuvent être de ce monde " |
./quelTypeTexte.sh
```

On peut rajouter l'option `-all` (uniquement lorsqu'on passe en paramètre le fichier) qui permet d'afficher les fréquences de toutes les types, elles correspondent au nombre de verbes (du texte) présents dans la liste des 20 verbes de chaque type, parmi les 100 verbes les plus présents du texte.

Commande de création de corpus : Soit corpus/typeTexte un dossier contenant des textes (.txt).

```
./apprendreTypeTexte.sh ../corpus/economie eco
./apprendreTypeTexte.sh ../corpus/philosophie philo
./apprendreTypeTexte.sh ../corpus/roman roman
...
```

Nous avons créé les types à partir des ouvrages suivant :

- **Philosophie** :
  - *Critique de la raison pure* - Emmanuel Kant
  - *Le Discours de la méthode* - René Descartes
  - *Écrits politiques* - Jean-Jacques Rousseau
  - *Métaphysique* - Aristote
  - *Pensées* - Blaise Pascal
  - *Lettres philosophiques* - Voltaire
  - *Système des Beaux-Arts* - Alain
- **Économie** :
  - *Le capital* - Karl Marx
  - *Manuel : comprendre l'économie* - Jacques Gouverneur
  - *Jacques Généreux explique l'économie* - Jacques Généreux
  - *Cours d'économie* - Charles Gide
  - *Principes d'économie* - Alfred Marshall
- **Roman** :
  Nous avons utilisé une compilation de ebooks, nous avons récolté environ *10000* ebooks au format .epub, puis nous avons mis en place un script permettant l'automatisation de la conversion de livres au format epub en fichiers txt interprétable par notre programme. Nous les avons utilisé pour faire le corpus de roman, nous en avons gardé quelques-uns pour faire des tests.

4 Résultats

Nous obtenons des résultats satisfaisants sur des textes de longueur correcte, en revanche les résultats sont plus mitigés pour des plus petites phrases.

4.0.1 Résultat de `quelleLangue` :

Voici quelques exemples de résultats obtenus après exécution de la commande `./quelleLangue.bash texte.txt -all` (on ne mettra ici que les trois premières langues) :

Texte	**Auteur**	**Résultat obtenu**
Un cruel hiver	Kate Sedley	fr : 77 ; ca : 25 ; nl : 14
Le 10 juin 1999 : la première guerre nucléaire vient de commencer	Robert Laffont	fr : 79 ; ca : 22 ; da : 14
Cycle d'Ogier d'Argouges	Pierre Naudin	fr : 78 ; ca : 22 ; da : 15

```
   ...quelleLangue$echo "phrase" > test.txt
...quelleLangue$ ./quelleLangue.bash test.txt -all
```

Résultats de la commande avec une phrase :

Phrase	**Résultat obtenu**
"Bonjour, cette phrase est en français, cela marche mieux si les phrases sont longues"	fr : 8 ; nl : 2 ; ca : 2 ;

On peut noter que malgré une phrase relativement petite (14 mots), on arrive à distinguer la langue relativement efficacement.

4.0.2 Résultat de `quelTypeTexte` :

Voici quelques exemples de l'utilisation du script `quelTypeTexte` (Il faut noter que les textes que nous utilisons pour nos tests ne sont pas des textes présents dans le corpus) :

Texte	**Auteur**	**Résultat obtenu**
Un cruel hiver	Kate Sedley	12 roman ; 6 philo ; 4 eco
L'être et le néant	Jean-Paul Sartre	9 philo ; 7 roman ; 6 eco
Commerce et emplois	Marion Jansen Eddy Lee	8 eco ; 3 philo ; 1 roman

5 Pour aller plus loin...

Nous avons fait le choix de créer un script global : `infotext.sh` permettant d'exécuter `quelleLangue.sh` et si le texte passé en paramètre est un texte français alors on exécute le script `quelTypeTexte.sh` qui indique le type du texte. A l'inverse, si la langue du texte n'est pas française alors il n'y a que la langue affichée en sortie. Bien que ce script soit assez simple il permet une utilisation plus aisée et rapide de `quelleLangue.sh` et `quelTypeTexte.sh`.

Le fait de se baser sur un plus gros corpus de texte pour la reconnaissance du type nous aurait permis de faire converger les 20 verbes vers un résultat plus précis.

Il est aussi possible d'utiliser le script `quelTypeTexte` pour trouver le type d'un texte d'une autre langue, pour cela il suffirait juste de créer un corpus à partir d'une autre langue sans effectuer de changements dans le script.

6 Conclusion

Ce projet nous a permis d'approfondir nos connaissances en bash, de créer un programme en prenant en compte l'optimisation des performances lors de traitements massif de données (traitement de plusieurs dizaines de giga-octets de fichiers textes).
Nous pouvons imaginer beaucoup de manières d'utiliser ces scripts, par exemple pour reconnaître un auteur il nous suffirait de remplacer nos corpus actuels (philosophie, économie et roman) par des corpus d'auteurs (par exemple : Hugo, Descartes, Rimbaud...).