Ingé4 - Travaux Pratiques de C++
TP1 : Calcul d'occurrences de lemme (binôme)

Johan Thapper
Polytech Paris-Sud, 2012-2013*

5 décembre 2012

1 Objectifs du TP

L'objectif de ce TP est de calculer les occurrences de mots dans un corpus avec les contraintes d'une structure de données imposée, d'un algorithme imposé, ainsi qu'une terminologie linguistique éventuellement non-familière.

Seront notamment abordés les points suivants :

- l'acquisition de données depuis un fichier texte et leur manipulation
- l'utilisation de structure de données : tableau associatif, tableau dynamique, objets
- l'écriture de données dans d'un fichier texte.
- l'utilisation des fichiers en-tête *map*, *string*, *fstream* et *vector*.

2 Rendu du TP

2.1 Archive

Archive au format tar.gz nommée `TP1_grpX_NOM1_NOM2.tar.gz` [1] (remplacez X par votre numéro de groupe, *NOM1* et *NOM2* par votre noms) à envoyer à `thapper@lri.fr`. La date limite, sans délai supplémentaire, est le 19 décembre. L'archive doit contenir un répertoire racine nommé `TP1_grpX_NOM1_NOM2` contenant :

- un répertoire `data` contenant `est-republicain-2002-TT.txt`, le corpus fourni
- un répertoire `bin` dans lequel est généré l'executable `occurences` par la commande *make all* du fichier `Makefile`
- un répertoire `src` contenant :
  - un fichier `main.cpp` lançant le programme

*Crédits du TP : M. Falco, P. Le Bodic

1. sans accents, `é` ↦ `e`, etc.

  - un fichier `Lemme.cpp` implémentant le fichier `Lemme.h`
- le fichier `Makefile` fourni et complété par vos soins pour correspondre au format de l'archive demandé
- un fichier `README.txt` expliquant votre rapport d'expérience sur ce TP : difficultés rencontrées, ce qui vous a plu/déplu, comment les structures de données peuvent être améliorées, comment l'algorithme peut être amélioré, compte-rendu sur les résultats du point de vue linguistique et du point de vue computationnel.

Les noms de variables doivent être explicites et utiliser le français, tout comme la documentation et les tâches d'expression écrite.

2.2 Barème prévisionnel

Une archive complète au format attendu, rendue dans les temps, qui compile et génère le fichier attendu obtiendra plus de douze. Les critères de notation sont :

- code qui ne compile pas : moins cinq points
- archive rendue en retard : moins cinq points
- archive ne répondant pas au cahier des charges (fichier manquant, nom de fichier différent) : moins trois points
- code non documenté, peu documenté ou trop documenté : moins trois point
- pas de fichier d'occurrences généré : moins trois point
- fichier d'occurrences généré différent des résultat attendus : pas forcément pénalisant (ça dépendra du type de l'erreur)

3 Corpus fourni

Le corpus provient du CNRTL [2] : plusieurs articles du journal *L'Est Républicain* ont été extraits du corpus de 2002 et centralisés en un seul fichier : `est-republicain-2002-TT.txt`. Chaque ligne dans ce fichier représente une phrase étiquetée et se trouve au format :

$[\text{ID}_1]\backslash t[\text{ID}_2]\backslash t[\text{mot}_1]\&[\text{étiquette}_1]\&[\text{lemme}_1] \ldots [\text{mot}_n]\&[\text{étiquette}_n]\&[\text{lemme}_n]$

où $[\text{ID}_1]$ est l'identifiant du document, $[\text{ID}_2]$ est l'identifiant de la phrase, \t signifie une tabulation (`'\t'` en C++), $[\text{mot}_i]$, $[\text{étiquette}_i]$, $[\text{lemme}_i]$ est un triplet et & signifie le caractère `'&'`.

L'*étiquette*, aussi appelée *tag* ou *pos*, correspond à la catégorie syntaxique du mot [3] ; elles ont été obtenues par le logiciel Treetagger [4] et peuvent se révéler parfois incorrectes : elles ne sont pas à corriger. Le *lemme* est la

2. Centre National de Ressources Textuelles et Lexicales, http://www.cnrtl.fr/corpus/estrepublicain/
3. *faire* est un verbe, *rouge* est un adjectif et un nom, *canard* est un nom
4. http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

forme non-fléchie d'un mot (entrée du dictionnaire : verbe à l'infinitif, nom et adjectif à la forme du masculin singulier) : le mot *programmons* a pour lemme *programmer*, le mot *lionnes* a pour lemme *lion*.

Nous allons comptabiliser les occurences des *lemmes*.

Fichiers à récupérer dans le répertoire http://www.lri.fr/~thapper/courses/cpp-2012/TP1 :

- `TP1.pdf`
- `est-republicain-2002-TT.txt`
- `Lemme.h`
- `Makefile`

4 Occurences des lemmes

Le but est de parcourir le corpus étiqueté pour compter les occurrences de chaque *lemme* autorisé à l'intérieur de chaque document tout en comptabilisant le nombre de documents différents le comportant.

- Créer un tableau dynamique en variable globale `vector<string> categories` contenant les six étiquettes autorisées des catégories suivantes : adjectif, adverbe, préposition, nom commun, nom propre et verbe.
- Créer un tableau associatif en variable globale `map<string, Lemme> tableLemme`
- Créer une fonction `vector<string> SegmenteSelonSymbole(string str, string symbole)` qui va découper `str` selon `symbole`.
- Créer une fonction `void CalculOccurrences(string nomDeFichier)` qui va :
  - parcourir `nomDeFichier` (le corpus fourni) en utilisant la fonction `SegmenteSelonSymbole`
    - premier découpage sur la ligne avec la tabulation pour obtenir l'identifiant du document, l'identifiant de la phrase et la phrase
    - deuxième découpage sur la phrase avec l'espace pour obtenir les lemmes
    - troisième découpage sur chaque lemme avec l'esperluette pour obtenir le mot, l'étiquette et le lemme
  - pour chaque lemme possédant une étiquette autorisée, s'il n'existe pas de clé ayant la valeur du lemme dans `tableLemme`, ajoutez-y le lemme en clé et créez un objet `Lemme` en valeur
  - si la clé existe :
    - s'il n'existe pas d'identifiant du document en clé de la donnée membre `tableOcc` de l'objet `Lemme`, ajoutez-la et incrémentez le nombre de documents où le lemme apparait.
    - si l'identifiant du document existe, incrémentez son nombre d'occurrences dans ce document.

5 Génération des occurrences dans un fichier texte

Écrire la fonction permettant de générer dans un fichier `resultats.txt` [5] les informations recueillies sur les occurrences ; une seule ligne par lemme, la ligne affichant les informations suivantes séparées par des tabulations :

- le lemme
- le nombre de documents possédant ce lemme
- le nombre d'occurrences totales de ce lemme

Ce fichier doit être généré dans le répertoire `data`. Quelques résultats attendus pour aider à la vérification du bon fonctionnement de votre programme :

```
avoir     905  3819
être      954  4411
étudiant  16   23
Rouge     3    3
rouge     21   29
abeille   2    3
```

5. Rappel : ce fichier n'est pas à rendre dans l'archive ; il doit pouvoir être généré à l'exécution par le programme.