TP 2
REQUÊTE SUR PLUSIEURS TEXTES

Introduction

Pour le premier tp2, vous devez construire un logiciel C++ qui répondra à une requête utilisateur. Un corpus de texte est fourni et l'utilisateur veut trouver les textes qui répondent le mieux à sa requête. Cette requête est composée de plusieurs mots et le logiciel doit trouver les textes qui contiennent ces mots. Cette recherche doit tenir compte du nombre d'occurrences d'un mot et de l'importance qu'a ce mot.

Description des entrées

Corpus

Une première entrée utilisée par votre logiciel est un corpus de texte. Ce corpus sera divisé en 12 fichiers contenant les textes et un autre fichier décrivant le corpus. Le fichier `listeDocument.xml` contient la description du corpus et les fichiers `janvier1930.xml` à `decembre1930.xml` contiennent les textes. Pour le reste de la description, nous utiliserons D_i pour désigner le $i^{ième}$ textes (ou histoire). Tous ces fichiers sont en format xml. Le code permettant la lecture de ces fichiers est déjà donné. Le code donné est composé de cinq fichiers :

- `DocumentXML.h` et `DocumentXML.cpp` : contiens le code permettant la lecture de fichier 'xml' (ce code est très très très simplifié et ne permet que de lire les fichiers du tp, ne fonctionnerait pas en général sur tout fichier xml).
- `Histoire.h` et `Histroire.cpp` : contient la classe `Histoire` qui encapsule un texte. Cette classe contient, entre autres, un itérateur (`begin, end`) qui parcourt tous les mots du texte. Elle contient aussi une méthode `titre` qui retourne le titre du texte.
- `inf3105.cpp` : contiens une fonction `lireDocuments` qui retourne un pointeur vers un vecteur d'histoire (`vector< Histoire *> *`). Cette fonction va lire tous les documents, séparer les textes et construire les `Histoire`s. Elle retourne le tout dans un vecteur, près pour l'utilisation. Ce fichier contient aussi un programme principal montrant l'utilisation de ces fonctions utiles pour votre logiciel.

Requête

Votre logiciel va demander une requête à l'utilisateur. Cette requête est une suite de mot. Un mot est une suite continue de caractère minuscule ou majuscule non accentué et possiblement le caractère '-'. Tout autre caractère sépare les mots. Pour le reste du texte, nous utiliserons t_j pour désigner le $j^{ième}$ mot de la requête. Votre logiciel répond à la requête et ensuite demande une autre requête. Tant que les requêtes contiennent des mots, votre logiciel répond à la requête et en demande une nouvelle. Lorsque la requête ne contient aucun mot, le logiciel va terminer.

Calcule

Votre logiciel va contenir deux phases de calcul. Une première lors du démarrage du logiciel et une seconde à chaque requête. La première phase va calculer des statistiques sur les mots des textes du corpus. Pour chaque histoire D_i vous devrez calculer le $\text{tf}(D_i, m)$ de chaque mot m contenu dans l'histoire. Pour ce calcul vous devez utiliser un arbre AVL qui contiendra les mots comme clef et le nombre d'occurrences de chaque mot comme définition. Le tf (terme frequency) d'un mot dans une histoire est simplement le nombre de fois que ce mot apparait dans la même histoire. Un mot aura donc un tf différent pour chaque histoire. Aussi, pour chaque mot, vous devrez calculer son $\text{idf}(m)$. L' idf (inverse document frequency) est inversement proportionnelle au logarithme du nombre d'histoires qui contiennent ce mot. Par exemple, pour un mot donné, s'il apparait dans 3 des 74 histoires, alors il aura un idf de $\log_2(74/3)$. Donc, si vous avez un mot m qui apparait dans x des N histoires, vous avez $\text{idf}(m) = \log_2(N/x)$. Vous devez aussi utiliser un arbre AVL pour ce calcul.

Dans la deuxième phase du logiciel, vous devez demander une requête R composée de mots à l'utilisateur. Cette requête est composée de k mots $\in R$. Ensuite, à l'aide des mots de la requête, vous devez évaluer une métrique $v(R, D_i)$ pour chaque histoire D_i.

$$v(R, D_i) = \sum_{j=1}^{k} \text{tf}(D_i, t_j) \times \text{idf}(t_j)$$

Lorsque vous avez calculé cette métrique pour chaque histoire, il reste à trouver les cinq histoires ayant la plus haute valeur et afficher leurs titres.

Production des sorties

Votre logiciel doit simplement afficher le titre des cinq meilleurs histoires précédés de leur métrique. Afficher un résultat par ligne (donc cinq lignes au total).

Directive

1. Le tp est à faire seul ou en équipe de deux (maximum).
2. Commentaire :

   a. Commentez l'entête de chaque fonction. Ces commentaires doivent contenir la description de la fonction et le rôle de ces paramètres.
   b. Une ligne contient soit un commentaire, soit du code, pas les deux.
   c. Utilisez des noms d'identificateur significatif.
   d. Utilisez le français.
3. Code :
   a. Pas de `goto`, `continue`.
   b. Les `break` ne peuvent apparaître que dans les `switch`.
   c. Un seul `return` par méthode.
4. Indentez votre code. Assurez-vous que l'indentation est faite avec des espaces.

Remise

Remettre le tp par l'entremise de Moodle. Vous devez remettre un fichier compressé du projet en utilisant exactement la méthode suivante sur le serveur Malt :

Dans le répertoire de votre projet, où seulement les fichiers `.h` et `.cpp` sont présent,

- `tar cvf tp2.tar *`
- `gzip tp2.tar`

Cela va produire un fichier `tp2.tar.gz` que vous allez remettre par l'entremise de Moodle.

Le tp est à remettre avant le 10 avril 23:59.

Évaluation

- Fonctionnalité (14 pts) : Votre tp doit compiler sans erreur (il peut y avoir des warnings). J'utilise la ligne de compilation suivante : `g++ *.cpp -lm -std=c++11`
- Structures (3 pt) : Il faut avoir **plusieurs** fonctions. Construisez un code bien structuré.
- Lisibilité (3 pts) : commentaire, indentation et noms d'identificateur significatif.