

## TD/TP 3 - TDA ARBRE

### Application au classement automatique de textes

#### 1 Introduction - Similarité de textes

On étudie ici une application du TDA arbre issue du domaine de la fouille de données (*data mining*). Il s'agit de la fouille de textes : disposant d'un ensemble de textes (articles, pages web, etc), on veut arriver à regrouper ces textes de façon automatique en fonction du thème auquel il se raccroche (musique, informatique, jeux, anti-libéralisme, etc).

Pour cela, on étudie la co-occurrence de mots et de concepts dans les textes à classer. On cherche à générer une matrice dont les lignes représentent les mots et/ou les concepts, et les colonnes représentent les textes à classer. Chaque case contient une valeur égale au nombre de fois où le mot (ou le concept) de la ligne intervient dans le texte de la colonne correspondante.

Une fois générée, cette matrice sera analysée par des classifieurs automatiques (ie, des algorithmes) qui vont proposer une répartition des documents initiaux en différents groupes suivant leur proximité sémantique. Plus deux textes ont des profils (colonnes) se ressemblant (ie, les mêmes mots utilisés dans des proportions proches), plus ils sont apparentés.

Une difficulté est que d'un texte à l'autre, il est possible qu'une même notion soit désignée par des mots différents mais synonymes (ex : "chanson" ou "morceau") ou par des mots plus ou moins spécialisés ("encodage" ou "mp3"). Ainsi une comparaison mot à mot des textes de départ ne suffit pas pour les classer correctement. Pour prendre en compte cette difficulté, une technique consiste à repérer dans les textes non seulement des mots mais aussi des *concepts* apparaissant implicitement à travers d'autres mots. Pour repérer le fait qu'un même concept apparaît dans des textes différents avec un des mots différents, on utilise des taxonomies (hiérarchie) de concepts, chacune liée à un domaine particulier. Chaque taxonomie a la forme d'un arbre, ce qui constitue pour nous le lien avec le TDA arbre.

#### 2 Le TDA Arbre en Java

On donne ci-dessous l'interface `Arbre.java` et l'implémentation (incomplète) `ArbreChaine.java` :

....

....

....

Le but des prochaines questions est de compléter la classe `ArbreChaine.java`.

**Question 1** Proposez une implémentation en Java des méthodes de l'interface manquantes dans l'implémentation.

**Question 2** Proposez une méthode `Noeud contient (Object obj)` qui renvoie le noeud auquel apparaît l'objet `obj` dans l'arbre implicite (`this`) ou renvoie `null` sinon.

**Question 3** Proposez une méthode `boolean descendant (Object m, Object c)` qui renvoie vrai ssi dans l'arbre implicite (`this`) l'objet `m` est situé dans un noeud descendant du noeud contenant l'objet `c`.

**Question 4** Proposez une méthode `int ancestralite (Object o1, Object o2)` qui renvoie 0 si les deux objets ne sont pas ancêtre l'un de l'autre dans l'arbre implicite (`this`) et qui renvoie sinon le nombre d'arêtes séparant les deux objets dans cet arbre.

Imaginons que chaque objet stocké dans l'arbre est un `Concept`, celui-ci ayant un attribut `String mot` (énonçant le concept) et un attribut `int generalite` désignant à quel degré de détail se situe ce concept (du plus particulier, `MaxInt`, au plus universel, 0).

**Question 5** Proposez une méthode `Void evaluateConcepts ()` qui étant donné un arbre d'objets `Concept`, remplit l'attribut `int generalite` de chaque concept en lui attribuant une valeur égale au nombre d'arêtes qui le sépare de la racine.

**Question 6** Quelle autre possibilité aurait pu être envisagée pour stocker la généralité des concepts dans une structure de donnée indépendante de l'arbre ?

### 3 Génération de la matrice d'occurrences des mots et des concepts

L'occurrence directe d'un mot dans un texte  $T$  est définie comme la fréquence de ce mot dans le texte, c'est-à-dire le nombre de fois où ce mot apparaît dans le texte divisé par le nombre de mots du texte. On note  $occ(m, T)$  ce nombre.

**Question 7** Etant donné un texte dont les mots sont stockés dans une liste, indiquer dans quelle(s) structure(s) vous pouvez stocker le nombre d'occurrences de chaque mot du texte.

**Question 8** *Ecrivez un algorithme qui calcule et stocke la valeur  $occ(m, T)$  pour tout mot  $m$  d'un texte  $T$ .*

Etant donné un texte  $T$  et un arbre de concepts  $A$ , l'**occurrence** d'un **concept**  $c$  de  $A$  dans  $T$  est notée  $occ(c, T)$ . Elle est évaluée suivant la formule suivante :

$$occ(c, T) = occ_m(c, T) + \sum_{m \in desc(c, A)} \frac{occ(m, T)}{gen(m, A)}$$

où

- $occ_m(c, T)$  désigne le nombre de fois où le mot  $c$  apparaît dans un texte  $T$  (un concept peut très bien être représenté implicitement par des mots dans le texte et apparaître lui aussi explicitement dans le texte !).
- $gen(c, A)$  désigne la generalite du concept  $c$  dans l'arbre  $A$  (définie comme ci-dessus).
- $desc(c, A)$  désigne l'ensemble des mots descendants de  $c$  dans l'arbre  $A$

**Question 9** *Etant donné une taxonomie de concepts représentée par un arbre et un texte  $T$ , indiquez dans quelle(s) structure(s) vous pouvez stocker les valeurs  $occ(c, T)$  pour les concepts  $c$  de la taxonomie.*

**Question 10** *Ecrivez un algorithme permettant d'obtenir l'occurrence des concepts d'un arbre dans un texte.*

**Question 11** *La présence de mots fonctionnels (les articles, les auxiliaires, les adverbess, etc) dans les textes analysés peut fausser les fréquences des mots et donc les calculs d'occurrences de mots et de concepts. Indiquez comment vous pouvez filtrer un texte pour le débarrasser de ses mots fonctionnels avant de lancer les calculs d'occurrence*

**Question 12** *Si on dispose maintenant de plusieurs textes (d'un tableau de listes) et de plusieurs arbres de concepts (un tableau d'arbres), quelle structure de données pouvez-vous utiliser pour stocker la matrice d'occurrences ?*

*On fera l'hypothèse qu'un concept n'apparaît que dans un seul arbre de concepts.*

**Question 13** *Modifiez l'algorithme précédent pour qu'il remplisse la matrice d'occurrences dans son ensemble.*