Algorithmes pour le Traitement de Données

Eric Gaussier
Myriam Tami, Thibaut Thonet

Univ. Grenoble Alpes
UFR-IM^{2}AG
eric.gaussier@imag.fr

2011

Table des matières

Objectifs du cours

- L'objectif de cours est d'introduire les principaux modèles et algorithmes utilisés en Traitement des Données
- Nous nous intéresserons en particulier à :
  - L'indexation et la représentation des documents
  - La recherche d'information (RI), y compris la RI sur le Web
  - La catégorisation et la classification (clustering)
  - La fouille de données

Domaines d'applications (1)

La recherche d'information (RI), y compris la RI sur le Web

- Indexation des requêtes et des documents
- Appariement entre requêtes et documents (ou entre documents)

Domaines d'applications (2)

La catégorisation

- Binaire vs multi-classe, mono- vs multi-label
- Hiérarchique ou à plat

Formalisation données étiquetées : $(x^{(i)}, y^{(i)}),\ 1 \leq i \leq n$ avec $x \in \mathbb{R}^p,\ y \in \mathcal{Y}\ \mathcal{Y} = \{-1; +1\}$; trouver une fonction f, de $\mathbb{R}^p$ dans $\mathcal{Y}$ qui à x associe y avec le moins d'erreurs possibles

Domaines d'applications (3)

La classification (clustering)

- Partitionnement vs affectation souple
- Hiérarchique ou à plat

Table des matières

Etapes de l'indexation textuelle

1. Segmentation
   - Découper un texte en mots :

     l'information donne sens à l'esprit
     l', information, donne, sens, à, l', esprit

     soit 7 mots mais seulement 6 types ; nécessité d'un dictionnaire (au moins léger) pour certains mots ou expressions
2. Filtrage par un anti-dictionnaire des mots vides
3. Normalisation
   - De la casse, des formes fléchies, des familles lexicales
   - Lemmatisation, racinisation

$\rightarrow$ Sac-de-mots : *inform, don, sens, esprit*

Représentation vectorielle des docs (1)

- L'ensemble des types forme le vocabulaire d'une collection. Soit M la taille de ce voc., et soit N le nombre de doc. dans la collection $\mathcal{C}$. On considére l'espace vectoriel à M dimensions ($\mathbb{R}^M$), dans lequel chaque axe correspond à un type
- Chaque document est alors représenté par un vecteur de $\mathbb{R}^M$ de coordonnées :
  - Présence/absence ou nbre d'occurrences du type dans le doc. : $w_i^d = \text{tf}_i^d$
  - Nombre d'occurrences normalisé par la long. du doc. : $w_i^d = \frac{\text{tf}_i^d}{\sum_{i=1}^M \text{tf}_i^d}$
  - Le *tf*idf* : $w_i^d = \frac{\text{tf}_i^d}{\sum_{i=1}^M \text{tf}_i^d} \underbrace{\log \frac{N}{\text{df}_i}}_{\text{idf}_i}$

    où df_i représente la fréquence documentaire du mot i : $\text{df}_i = \sum_{d \in \mathcal{C}} \mathbb{1}\{i \in d\}$

Représentation vectorielle des docs (2)

La représentation vectorielle adoptée permet d'avoir directement accès aux outils mathématiques associés : distances, similarités, réduction de dimensions, ...

Exercices

- Chaque document est représenté par un tableau à M dimensions contenant les poids (coordonnées) des termes (types, mots) ; écrire un algorithme qui calcule le produit scalaire entre 2 documents
- Quelle est la complexité d'un algorithme qui calcule le produit scalaire entre un document et tous les autres documents de la collection ?

Une représentation creuse !

La majorité des termes de la collection n'apparaissent pas dans un document donné ; chaque document a donc la majeure partie de ses coordonnées nulles ; un gain d'espace peut être obtenu en ne représentant pas ces coordonnées nulles

Exemple de représentation creuse :

document d {

int l	(long. du doc. (types))
TabTermes int[l]	(indices des termes, ordre croissant)
TabPoids float[l]	(poids des termes)
...	

Reprendre l'exercice précédent avec cette représentation

Le fichier inverse

Il est possible d'accélérer le calcul du produit scalaire, dans le cas de représentations creuses, en utilisant un *fichier inverse*, c'est-à-dire une structure qui fournit pour chaque terme l'ensemble des documents dans lesquels il apparaît :

terme i {
- int l — (nbre de docs assoc.)
- TabDocs int[l] — (indices des docs, ordre croissant)
- ...

Reprendre l'exercice précédent avec cette structure

Remarque Avantageux avec toute mesure (distance, similarité) qui ne fait pas intervenir les termes non présents dans un document
Produit scalaire ?, cosinus ?, distance euclidienne ?

Construction du fichier inverse

Dans le cadre d'une collection statique, 3 étapes principales régissent la construction du fichier inverse :

1. Extraction des paires d'identifiants *(terme, doc)*, passe compléte sur la collection
2. Tri des paires suivant les id. de terme, puis les id. de docs
3. Regroupement des paires pour établir, pour chaque terme, la liste des docs

Ces étapes ne posent aucun probléme dans le cas de petites collections oï tout se fait en mémoire

Quid des grandes collections ?

Cas de mémoire insuffisante

Il faut dans ce cas stocker des informations temporaires sur disque
Trois étapes :

1. Collecte des paires TermId-DocId et écriture dans un fichier
2. Lecture par blocs de ce fichier, inversion de chaque bloc et écriture dans une série de fichier
3. Fusion des différents fichier pour créer le fichier inversé

Algorithme associé : *Blocked sort-based indexing* (BSBI)

L'algorithme BSBI (1)

1. $n \leftarrow 0$
2. while (tous les docs n'ont pas été traités)
3. do
4. $\quad n \leftarrow n + 1$
5. $\quad$ block $\leftarrow$ ParseBlock()
6. $\quad$ BSBI-Invert(block)
7. $\quad$ WriteBlockToDisk(block,f_n)
8. MergeBlocks(f_1, ..., f_n ;f_{merged})

L'algorithme BSBI (2)

L'inversion, dans BSBI, consiste en un tri sur les identifiants de termes (premiére clé) et les identifiants de documents (deuxiéme clé). Le résultat est donc un fichier inverse pour le bloc lu. Complexité en $O(T \log T)$ oï T est le nombre de paires terme-document (mais étapes de lecture et de fusion peuvent être plus longues)

Exemple

t_1 = "brutus", t_2 = "caesar", t_3 = "julius", t_4 = "kill", t_5 = "noble"

$t_1 : d_1$	$t_2 : d_4$	$t_2 : d_1$
$t_3 : d_{10}$	$t_1 : d_3$	$t_4 : d_8$
$t_5 : d_5$	$t_2 : d_2$	$t_1 : d_7$

Table des matières

Ouvrages de référence

- Recherche d'information : Applications, modèles et algorithmes - *M.-R. Amini, E. Gaussier* - Eyrolles, 2ième édition, 2017

- Introduction to Information Retrieval - *C. Manning, P. Raghavan, H. Schütze* - Cambridge Univ. Press, 2008