INFO 4 2025-2026

HMUL8R6B: Accès et recherche d'information

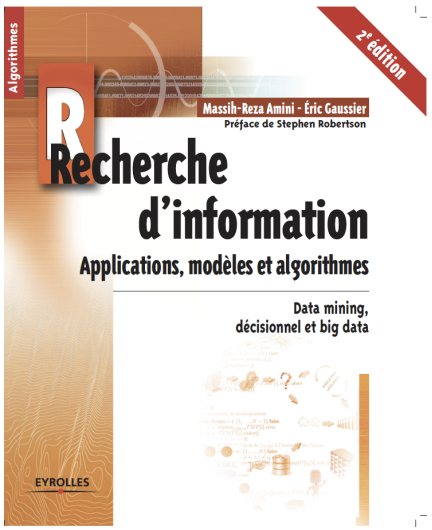
Indexation, représentation

Philippe Mulhem

Université Grenoble Alpes
Laboratoire d'Informatique de Grenoble
Philippe.Mulhem@imag.fr
https://hmul8r6b.imag.fr/doku.php

Objectifs du module

- L'objectif de cours est d'introduire les principaux modèles et algorithmes utilisés en Recherche d'Information (RI).
- Nous nous intéresserons en particulier à :
  1. L'indexation et la représentation des documents
  2. Les modèles standard de la RI
  3. La RI sur le web
- Projet : moteur de recherche texte
- Cours basé entre autre sur l'ouvrage :

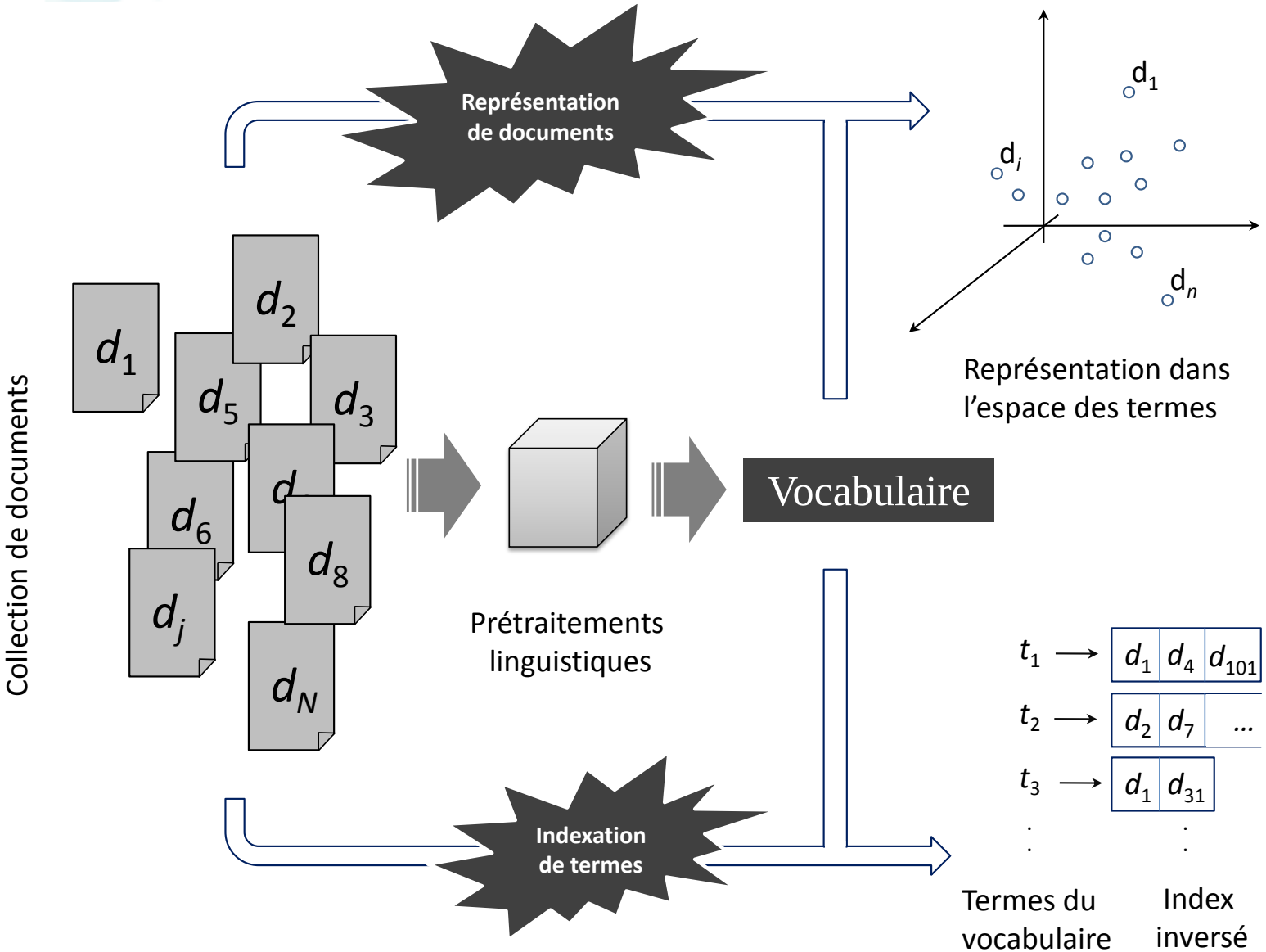


La Recherche d'Information (RI)

Objectif

Retrouver des **documents pertinents** pour un **utilisateur** qui a un **besoin d'information**, en se basant sur le **contenu** des documents.

Indexation et représentation



Prétraitements linguistiques : segmentation

Segmentation (en anglais *tokenisation*) consiste à séparer une suite de caractères en éléments sémantiques, ou *mots* :

l'information donne sens à l'esprit

Après segmentation $\Rightarrow$ *l', information, donne, sens, à, l', esprit*

soit 7 occurrences de mots mais seulement 6 mots ("l'" apparaît 2 fois).

Prétraitements linguistiques : segmentation

- Une bonne segmentation prend en compte des spécificités de la langue des textes traités.
- Pour certaines langues asiatiques comme par exemple le chinois, les mots dans un texte ne sont pas séparés par des espaces, la segmentation est alors une tâche ardue.
- Pour des langues indo-européennes la tâche de segmentation est plus aisée puisque *l'espace*, et *les signes de ponctuation* donnent une première indication de séparation entre les différents éléments lexicaux. Néanmoins, chaque langue de ce groupe linguistique a sa spécificité propre.

Prétraitements linguistiques : segmentation

- Cas du français
  - Les composés lexicaux à trait d'union comme *chassé-croisé*, *peut-être*, *rendez-vous*, etc.
  - Les composés lexicaux à apostrophe comme *jusqu'où*, *aujourd'hui*, *prud'homme*, etc.
  - Les expressions idiomatiques comme *au fait*, *poser un lapin*, *tomber dans les pommes*, etc.
  - Les formes contractées comme *M'sieur*, *j'*, etc.
  - Les sigles et les acronymes comme *K7, A.R., CV, càd, P.-V.*, etc.
- Ce problème devient même extrême avec l'allemand, où les noms composés s'écrivent sans espace; par exemple : *Rindfleischetikettierungsüberwachungsaufgabenübertragungsgesetz*:

  (Loi de délégation de surveillance de l'étiquetage de la viande bovine)

Prétraitements linguistiques : segmentation

- Pour les langues européennes, différents logiciels de segmentation existent.
- Certains de ces logiciels font une analyse plus poussée du texte en associant aux mots d'une phrase leur fonction grammaticale (ex.: *TreeTagger*, libre et couramment utilisé pour cette tâche : `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`).
- La segmentation dans ces cas est une étape préliminaire à cette analyse.

Prétraitements linguistiques : normalisation

- La *normalisation de mots* est le processus qui transforme tous les mots d'une même famille sous une forme *normale* ou *canonique* de façon à ce que l'appariement entre les termes d'une requête et ceux de l'index puissent avoir lieu, et ce malgré les différences possibles entre les séquences de caractères de leurs mots associés.

  moteur et *moteurs* seront mis sous même forme normale

- Les règles de normalisation varient suivant les deux types de normalisation *surfacique* ou *textuelle* et normalisation *linguistique*

Prétraitements linguistiques : normalisation textuelle

La normalisation textuelle rend les mots d'une même famille sous leur forme canonique en effectuant quelques transformations superficielles sur les séquences de caractères de ces mots.

- **Les ponctuations.** La règle de base appliquée est d'enlever les points et les traits d'union apparaissant dans les mots.
- **La casse.** Une stratégie classique est de réduire les lettres en minuscules. Cette stratégie peut rater avec certains noms propres (`Apple` et `apple` en anglais.)
- **Les accents.** La règle appliquée consiste généralement à enlever les diacritiques sur tous les mots (par exemple *ambiguë* devient *ambigue* ou *forêt* qui devient *foret*).

Prétraitements linguistiques : normalisation linguistique

La normalisation linguistique consiste à ramener un mot fléchi sous sa forme canonique. 2 approches classiques :

- La *racinisation* qui regroupe les différentes variantes morphologiques d'un mot autour d'une même *racine* (en anglais *stem*). Ce procédé repose sur un ensemble de règles pour supprimer les flexions et les suffixes des mots. Il est fortement dépendant de la langue. `http://snowball.tartarus.org/algorithms/french/stemmer.html`. Un exemple connu : Algorithme de Porter.
  Pour l'anglais 4 règles par ex. tirées de Porter :
  1. s $\rightarrow$ / 2. e $\rightarrow$ / 3. ies $\rightarrow$ y 4. ss $\rightarrow$ s

- La *lemmatisation* fait une analyse linguistique poussée pour enlever les variantes flexionnelles des mots afin de les ramener sous leur forme *lemmatisée* ou encyclopédique.

Prétraitements linguistiques : Filtrage par anti-dictionnaire

- Le filtrage supprime les mots présents avec une fréquence élevée dans les documents d'une collection et qui apportent peu d'information sur le contenu d'un document.
- Les 42 mots les plus fréquents (et peu informatifs) présents dans la collection de Wikipédia français 2013 :

de	la	le	et	en	l'	du
des	d'	les	est	un	une	il
au	dans	par	pour	sur	date	a
qui	que	avec	son	plus	se	sans
quel	quelle	s'	pas	n'	je	y
ou	se	sont	aux	qu'	sa	elle

→ Sac-de-mots après racinisation + anti-dictionnaire :
inform, don, sens, esprit

Quelques statistiques sur la collection Wikipédia français 2013

Variables	Symboles	Valeurs
# de documents de la collection	N	1 349 539
# total d'occurrences des mots	M	696 668 157
# moyen de mots par document		416
Taille de la collection ségmentée sur le disque		4.6 GB
# de mots	M_y	757 476
# de mots après racinisation	M_{Nor}	604 444
# de termes du vocabulaire (antidico ôté)	V	604 244
# moyen de termes par document		225
Taille de la collection prétraitée sur le disque		2.8 GB

Les deux lois de base en RI

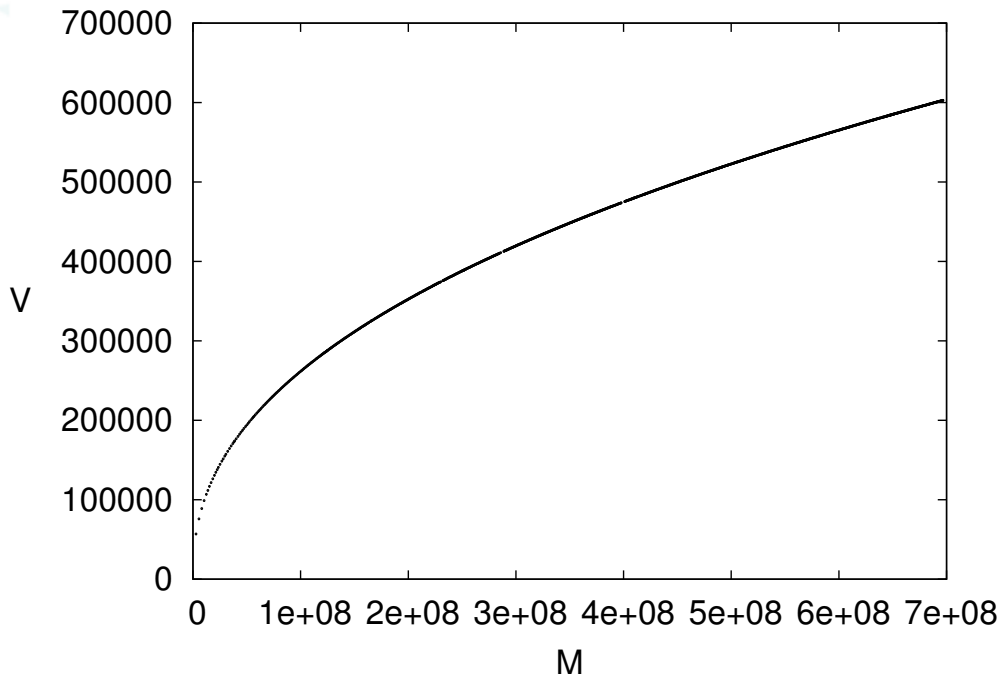
- On remarque d'abord la grande taille du vocabulaire de la collection. Le Petit Robert contient 60 000 entrées et le Grand Robert en contient 75 000, un relevé le plus exhaustif (wiktionary.org) montre que la langue française contiendrait à peu près 550 000 mots. Grand nombre de mots différents (757 476) dans le Wikipédia français
- Le filtrage des documents par l'anti-dictionnaire constitué des 200 mots vides réduit le nombre moyen d'occurrence de mots dans les documents de 416 à 225 : près de 45% des occurrences de mots dans les textes de Wikipédia sont les 200 mots les plus fréquents de la collection. De plus leur filtrage allège l'espace disque de près de 39% (de 4,6 GB à 2,8 GB).

Loi de Heaps

La loi de Heaps stipule que la taille du vocabulaire (V) croît exponentiellement en fonction du nombre de mots présents dans une collection donnée (M).

$$V = K \times M^{\beta}$$

avec K $[10, 100]$, et $\beta \in [0.4, 0.6]$.



Loi de Zipf

La loi de Zipf spécifie que la fréquence d'occurrence fc(m) d'un mot m dans une collection de documents est inversement proportionnelle à son rang :

$$\forall m, \text{fc}(m) = \frac{\lambda}{\text{rang}(m)}$$

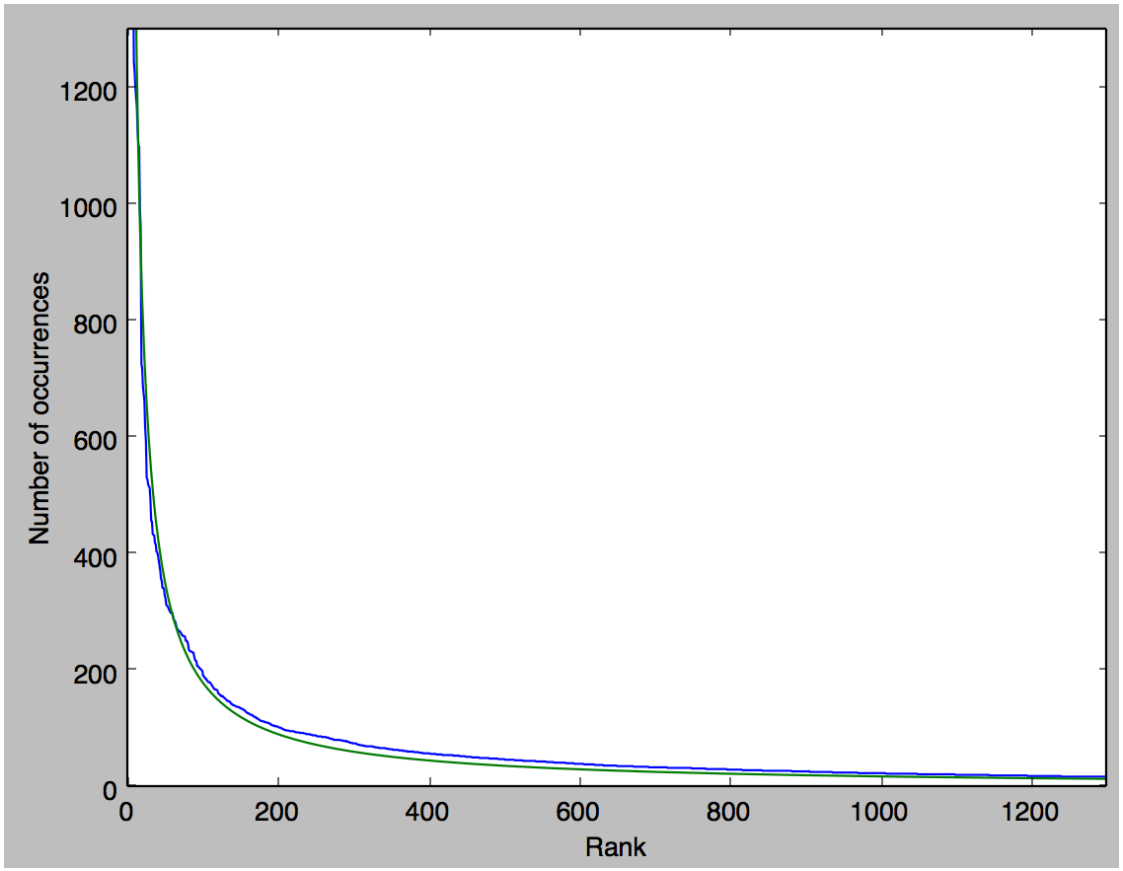
Cette valeur est une mesure théorique fc_{the}.
On peut compter expérimentalement ces fréquences fc_{exp} :

rang	mot	fc_{exp}
1	*de*	36,875,868
2	*la*	16,565,726
3	*le*	12,639,034
4	*et*	11,587,487
5	*en*	10,885,221
6	*l'*	8,937,203
7	*du*	8,541,846
8	*des*	8,302,026

Loi de Zipf

Il est possible d'estimer théoriquement la valeur λ par $\frac{M}{ln(M_y)}$.

La courbe réelle (en bleu) et théorique (en vert) sur les données du TP est :



Loi de Zipf

On passe des fréquences à des probablilités par : $p(m) = \frac{fc(m)}{M}$

Représentation vectorielle des docs

- Le modèle vectoriel (ou *Vector Space Model*), proposé par Gerard Salton, définit la représentation documentaire communément utilisée dans différentes tâches de l'accès à l'information.
- Avec cette représentation on associe à chaque document d d'une collection $\mathcal{C}$, un vecteur $\mathbf{d}$, dont la dimension correspond à la taille du vocabulaire V. L'espace vectoriel considéré est donc un espace vectoriel de termes dans lequel chaque dimension est associée à un terme de la collection.

$$\forall d \in \mathcal{C}, \mathbf{d} = (w_{id})_{i \in \{1,\ldots,V\}}$$

Pondération des termes

Elle repose sur 3 critères, pour un terme t_i et un document d :

- caractérisation du terme dans le document : $p_{\text{tf}_{t_i,d}}$
- caractérisation du terme dans le corpus : $p_{\text{df}_{t_i}}$
- une normalisation du vecteur qui dépend du document : n_d

Le poids d'un terme t_i dans un document d est défini par :

$$\forall d \in \mathcal{C}, \forall i \in \{1, .., V\}, w_{id} = p_{\text{tf}_{t_i,d}} \times p_{\text{df}_{t_i}} \times n_d$$

Pondérations de termes les plus usuelles

$$\forall d \in \mathcal{C}, \forall i \in \{1, .., V\}, w_{id} = p_{\mathrm{tf}_{t_i,d}} \times p_{\mathrm{df}_{t_i}} \times n_d$$

$p_{\mathrm{tf}_{t_i,d}}$	$p_{\mathrm{df}_{t_i}}$	n_d
$\mathrm{tf}_{t_i,d}$	1	1
$\begin{cases} 1 & \text{si } \mathrm{tf}_{t_i,d} > 0, \\ 0 & \text{sinon.} \end{cases}$	$\mathrm{idf}_{t_i} = \ln \frac{N}{\mathrm{df}_{t_i}}$	$\frac{1}{\|d\|} = \frac{1}{\sqrt{\sum_{i=1}^{V} w_{id}^2}}$
$\frac{1+\ln(\mathrm{tf}_{t_i,d})}{1+\ln(moy\_tf(d))}$		
$0.5 + 0.5 \times \frac{\mathrm{tf}_{t_i,d}}{\mathrm{tf}_{\max_d}}$	$\max\{0, \ln \frac{N-\mathrm{df}_{t_i}}{\mathrm{df}_{t_i}}\}$	$\frac{1}{(Char_d)^\alpha}, 0 < \alpha < 1$
$\begin{cases} 1 + \ln(\mathrm{tf}_{t_i,d}) & \text{si } \mathrm{tf}_{t_i,d} > 0, \\ 0 & \text{sinon.} \end{cases}$		

Dans le projet, vous utilisez : $p_{\mathrm{tf}_{t_i,d}} = tf_{t_i,d}$, $p_{\mathrm{df}_{t_i}} = \mathrm{idf}_{t_i}$, et $n_d = 1$.

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 V dimensions contenant les poids (coordonnées) des termes (ex. : 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 d'une collection de N documents ?

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
- Le codage par *indice-valeur* consiste à obtenir une représentation compacte des vecteurs de documents en ne codant que leurs caractéristiques non nulles ainsi que les indices associés à ces caractéristiques.
  Par exemple, le codage par indice-valeur du vecteur

$$\vec{v} = (0,0 \quad 0,1 \quad 0,0 \quad 0,0 \quad 4,9 \quad 0,0 \quad 0,0 \quad 1,3 \quad 0,0)$$

est:

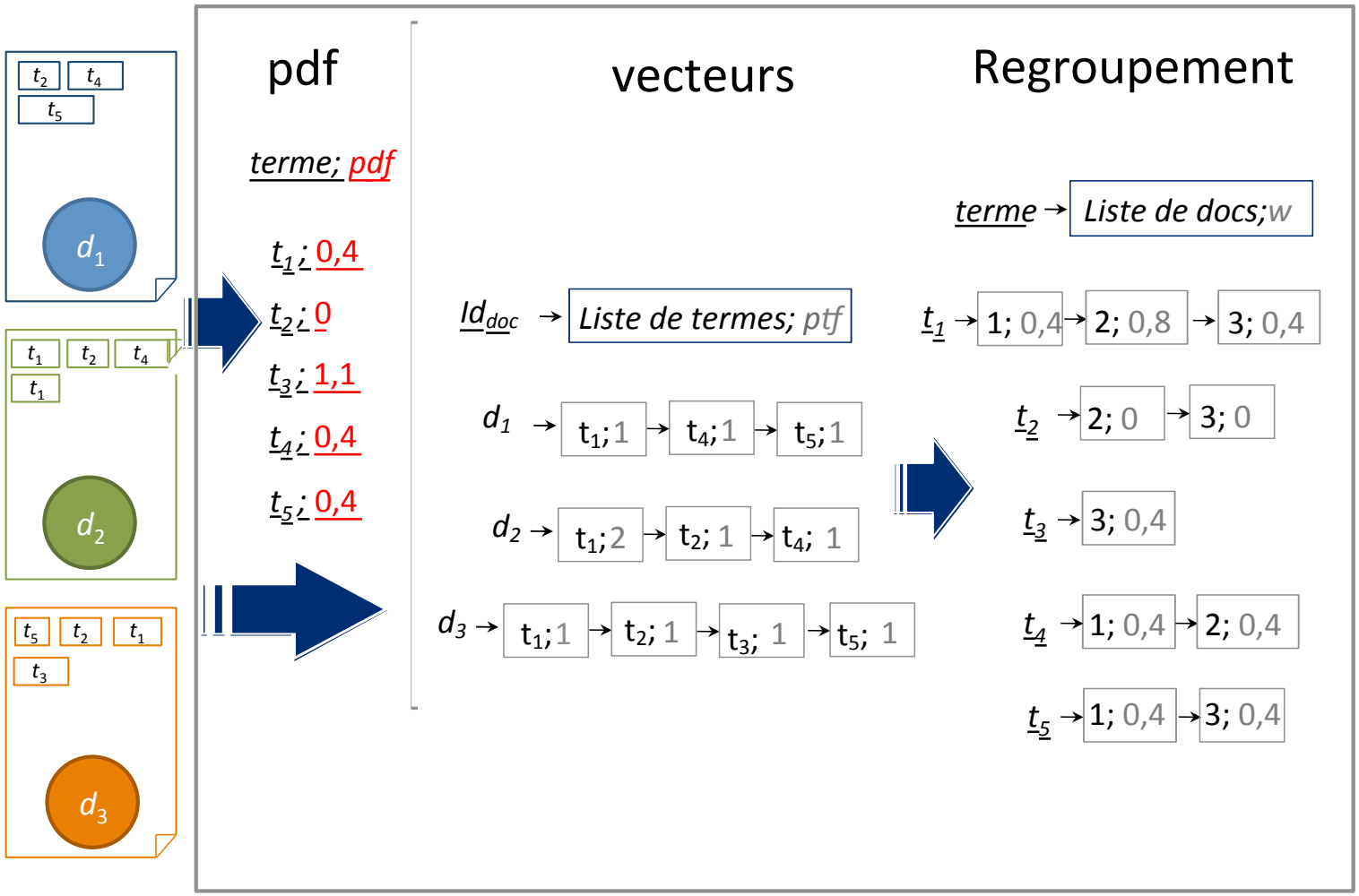
$$\vec{v}_{\text{ind-val}} = (2:0,1 \quad 5:4,9 \quad 8:1,3)$$

Reprendre l'exercice précédent avec cette représentation

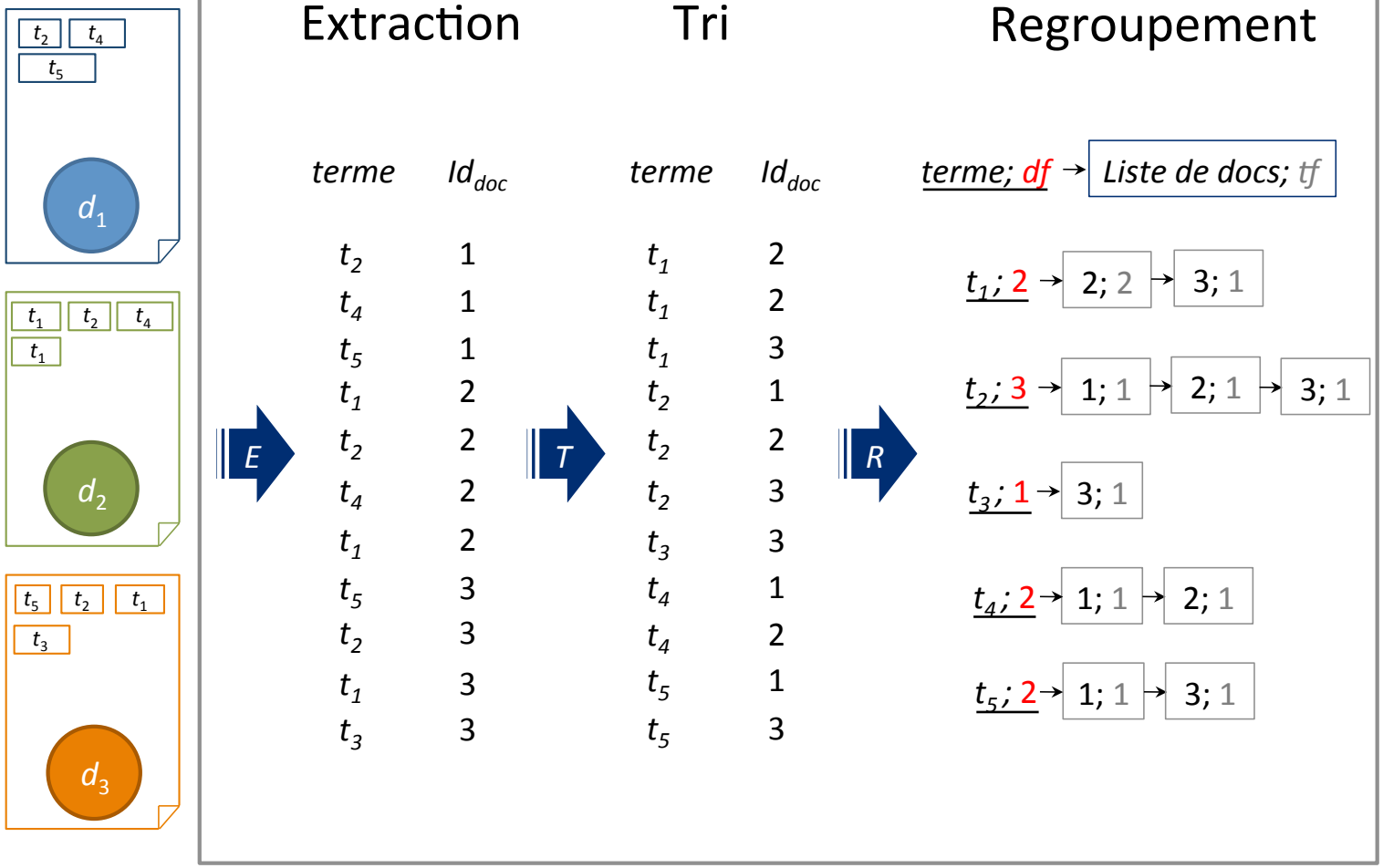
Index inversé

- La structure de données qui fait correspondre chaque terme du vocabulaire à la liste des documents qui le contiennent est la façon la plus rapide pour trouver un terme d'une requête donnée dans une collection de documents. Cette structure est appelée index inversé.
- On peut construire cet index inversé à partir des vecteurs (cas du TP), ou bien avec un processus qui évite de tout stocker en mémoire (voir la suite)

Construction de l'index inversé : à partir des vecteurs



Construction de l'index inversé : sans vecteur



Construction de l'index inversé : sans vecteur

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 complète 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 problème dans le cas de petites collections où tout se fait en mémoire

Quid des grandes collections ? Stockage en RAM impossible...

Index inversé: cas distribué