RICM 4

HMUL8R6B: Accès et recherche d'information

Indexation, représentation

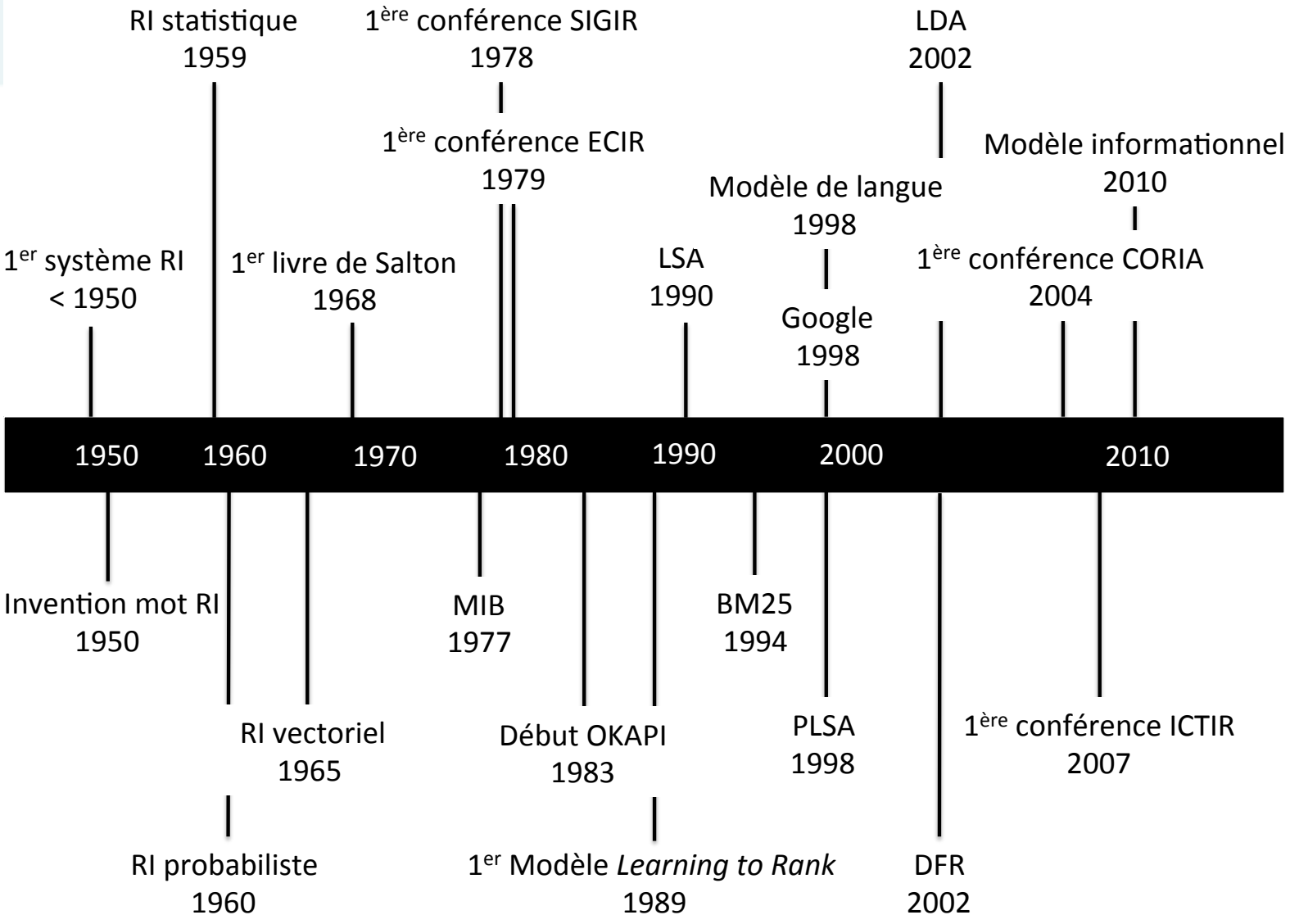
Massih-Reza Amini

Université Joseph Fourier
Laboratoire d'Informatique de Grenoble
Massih-Reza.Amini@imag.fr

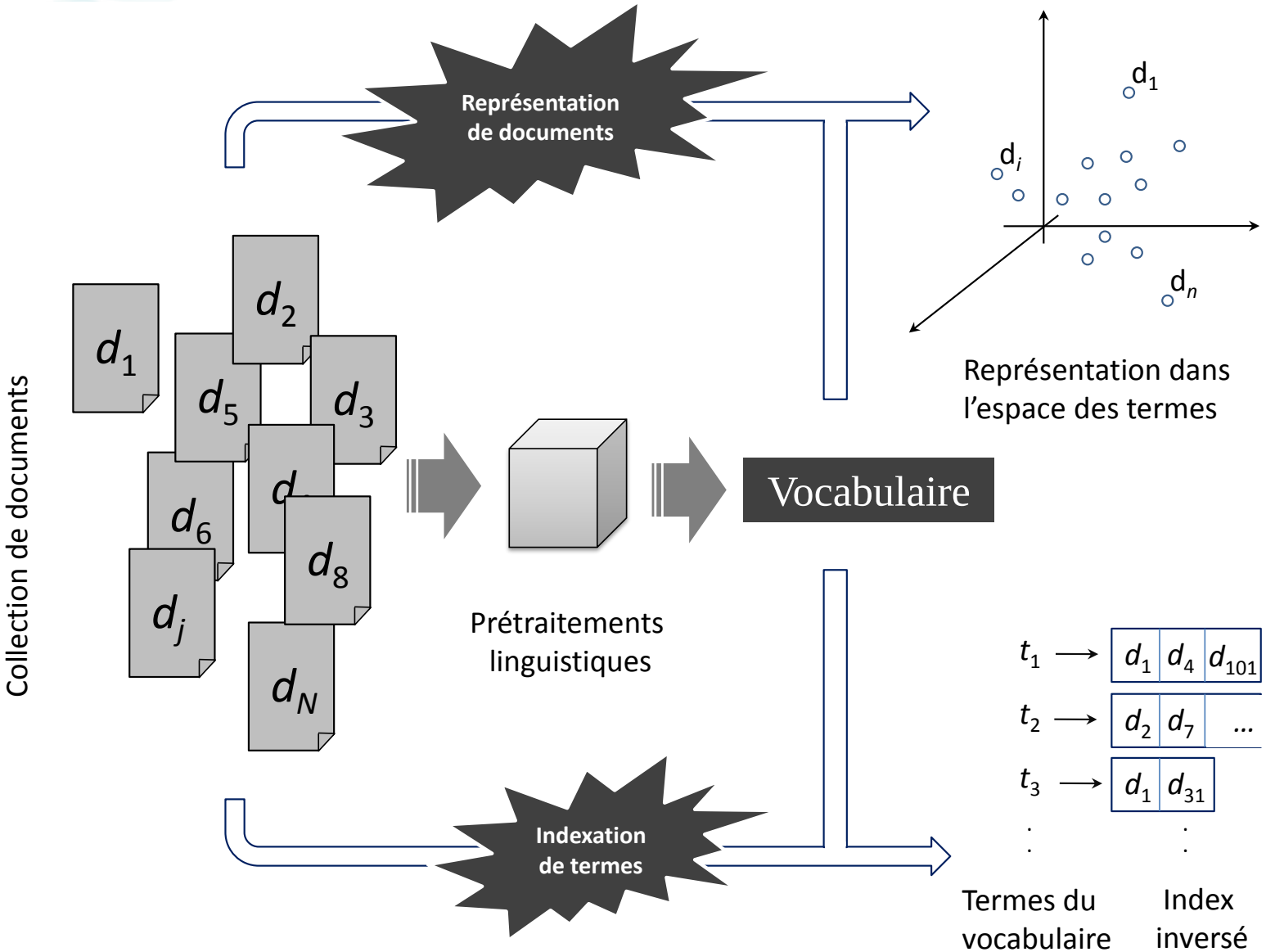
Objectifs du module

- 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 modéles standard de la RI
  3. La RI sur le web
- Projet : moteur de recherche texte
- Cours basé sur l'ouvrage :

Les faits marquants de l'évolution de RI



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 mots mais seulement 6 types.

Un type de mot est la classe de tous les mots ayant la même séquence de caractères.

Prétraitements linguistiques : segmentation

- Une bonne segmentation dépend de la prise 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 dans ce cas 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

- Par exemple pour le français, nous avons
  - 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 *Gad'zarts (les gars des Arts et Métiers)*, *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

Prétraitements linguistiques : segmentation

- Pour les langues européennes, différents logiciels de segmentation commerciaux existent.
- Le but de certains de ces logiciels est de faire une analyse plus poussée du texte en associant aux mots d'une phrase leur fonction grammaticale.
- La segmentation dans ces cas est une étape préliminaire à cette analyse. On peut citer ici le logiciel *TreeTagger* qui est en libre accès et est couramment utilisé pour cette tâche : `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`.

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 qui peut y avoir entre les séquences de caractères de leurs mots associés.

  P.V., P.-V. et PV 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 à l'exemple ci-dessus, et qui concerne tous les acronymes, est d'enlever les points et les traits d'union apparaissant dans les mots.
- **La casse.** Une stratégie classique est de réduire certaines lettres en minuscules. Cette stratégie ne peut pas se généraliser à cause de certains noms propres (`Blanc`, `Noir`) et des acronymes.
- **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.

- La *racinisation* consiste à regrouper 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, et il est fortement dépendant de la langue utilisée.
  `http://snowball.tartarus.org/algorithms/french/stemmer.html`

- 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

- Le filtrage supprime les mots qui tendent à être présents avec une fréquence élevée dans tous les documents d'une collection et qui n'apportent que 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 :

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 : *inform, don, sens, esprit*

Quelques statistiques sur la collection de Wikipédia français prétraitée

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 types de mots	M_y	757 476
# de types de mots après racinisation	M_{Nor}	604 444
# de termes du vocabulaire	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

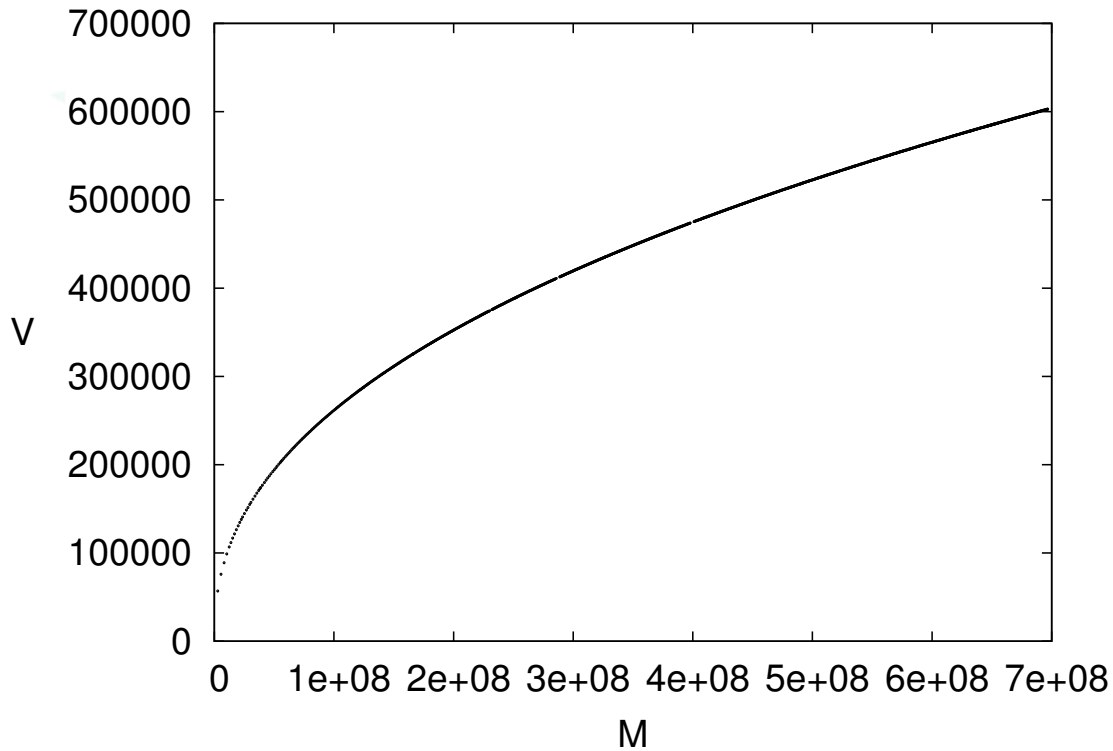
- On remarque d'abord la grande taille du vocabulaire de la collection. Le Petit Robert contient 60, 000 entrées et le Grand Robert en contient 75, 000, le relevé le plus exhaustif montre que la langue française contiendrait à peu près 700, 000 mots. La question ici est alors de savoir pourquoi y-a-t-il un si grand nombre de mots différents (757, 476) dans le Wikipédia français?

- On note ensuite que le filtrage des documents par l'anti-dictionnaire, constitué des 200 mots vides, a réduit le nombre moyen d'occurrence de mots dans les documents de 416 à 225. Autrement dit, près de 45% des occurrences de mots dans les textes de Wikipédia correspond aux 200 mots les plus fréquents de la collection. Et on voit que 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).

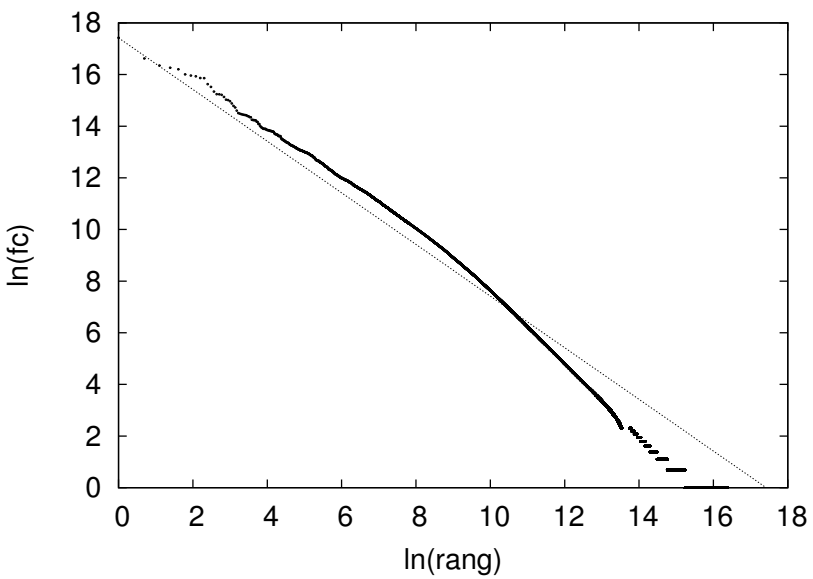
$$V = K \times M^{\beta}$$



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 :

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



rang	mot	fréquence
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

Exercices

On suppose que la présence d'un mot dans un document est le résultat du tirage aléatoire avec remise dans l'ensemble des M_y différents types de mots de la collection de documents.

- Soit X_k la variable aléatoire binaire correspondant au résultat de l'évènement E_k *: le* k^{ieme} *mot le plus fréquent a été tiré lors du tirage*. Montrer que $P(X_k = 1) = \frac{1}{k \times \ln(M_y)}$.

  Indication : On suppose que $\sum_{i=1}^{r} \frac{1}{i}$ *est approchée par* $\int_1^r \frac{dx}{x} = \ln r$

- Soit d un document de longueur n (en nombre de mots). On cherche à calculer le nombre moyen de fois où le k^{ieme} mot le plus fréquent de la collection apparaît dans d. Montrer que la somme S_k de n variables $X_{k,i}$, $i \in \{1, ..., n\}$, indépendantes, exprimant chacune le résultat de l'évènement E_k, suit une loi binomiale dont on précisera les paramètres. En déduire l'espérance de S_k.
- Pour un document de la collection Wikipédia, quel serait le nombre moyen d'apparitions du mot le plus fréquent, *de*, dans un document de taille 416 mots ? Comparer ce résultat avec le nombre moyen d'apparitions observées de *de* dans un document de la collection (27).

Exercices

On suppose que le 50^{ieme} mot le plus fréquent d'une langue ℓ donnée a une probabilité d'apparition de 0.02 dans n'importe quelle collection de cette langue. Soit $\mathcal{C}_\ell$ un corpus de documents de la langue ℓ contenant $10,000$ mots différents.

- Quel est le rang d'un mot qui apparaît 40 fois dans $\mathcal{C}_\ell$?
- Dans $\mathcal{C}_\ell$, combien de mots ont une fréquence d'apparition égale à 10 ?
- Pour une collection de documents avec une constante de Zipf égale à λ, quel est le dernier rang r à partir duquel des mots différents peuvent avoir une même fréquence ?

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. L'espace vectoriel considéré est donc un espace vectoriel de termes dans lequel chaque dimension est associée à un terme de la collection.

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

Pondérations de termes les plus usuelles

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

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

Exercice : C'est quoi l'idf ?

L'*entropie* de Shannon d'une source, représentée par une variable aléatoire (*v.a.*) X prenant b valeurs différentes, est la quantité d'information contenue ou délivrée par cette source. Formellement, sur un ensemble $\mathcal{S} = \{x_1, \ldots, x_n\}$ contenant n observations qui sont chacune une réalisation de X, cette entropie est estimée par:

$$H_b(X) = -\sum_{i=1}^{n} P(X = x_i) \log_b \left(P(X = x_i)\right)$$

Dans la suite de cet exercice on ne considérera que le logarithme népérien, puisque tout logarithme d'une base quelconque $b \in \mathbb{R}_+^*$ peut s'obtenir en fonction de ce dernier.

Soit $\mathcal{C} = \{d_j\}_{j=1}^{N}$ une collection de N documents, où chaque document $d_j \in \mathcal{C}$ est la réalisation d'une variable aléatoire D suivant une loi de probabilité uniforme. Montrer dans ce cas que l'entropie de D est :

$$H(D) = -\ln \frac{1}{N}$$

Dans le cas où l'on dispose de deux sources X et Y, l'entropie restante provenant de la *v.a.* X lorsque l'on connaît parfaitement Y, s'appelle *l'entropie conditionnelle*, $H(X \mid Y)$, et est calculée par :

$$H(X \mid Y) = -\sum_{j=1} P(X = x_j \mid Y) \ln \Big(P(X = x_j \mid Y)\Big)$$

On suppose que les termes du vocabulaire $\mathcal{V} = \{t_i\}_{i=1}^{V}$ associé à la collection $\mathcal{C}$ sont des réalisations d'une *v.a.*, T, et on considère le sous-ensemble $\mathcal{C}_{t_i} \subseteq C$ contenant df_{t_i} documents dont chacun contient le terme $t_i \in \mathcal{V}$. Dans le cas où les documents de $\mathcal{C}_{t_i}$ suivent une loi uniforme, exprimer l'entropie conditionnelle $H(D \mid T = t_i)$ en fonction de $\mathrm{df}(t_i)$ et de idf_{t_i}.

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 (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 :

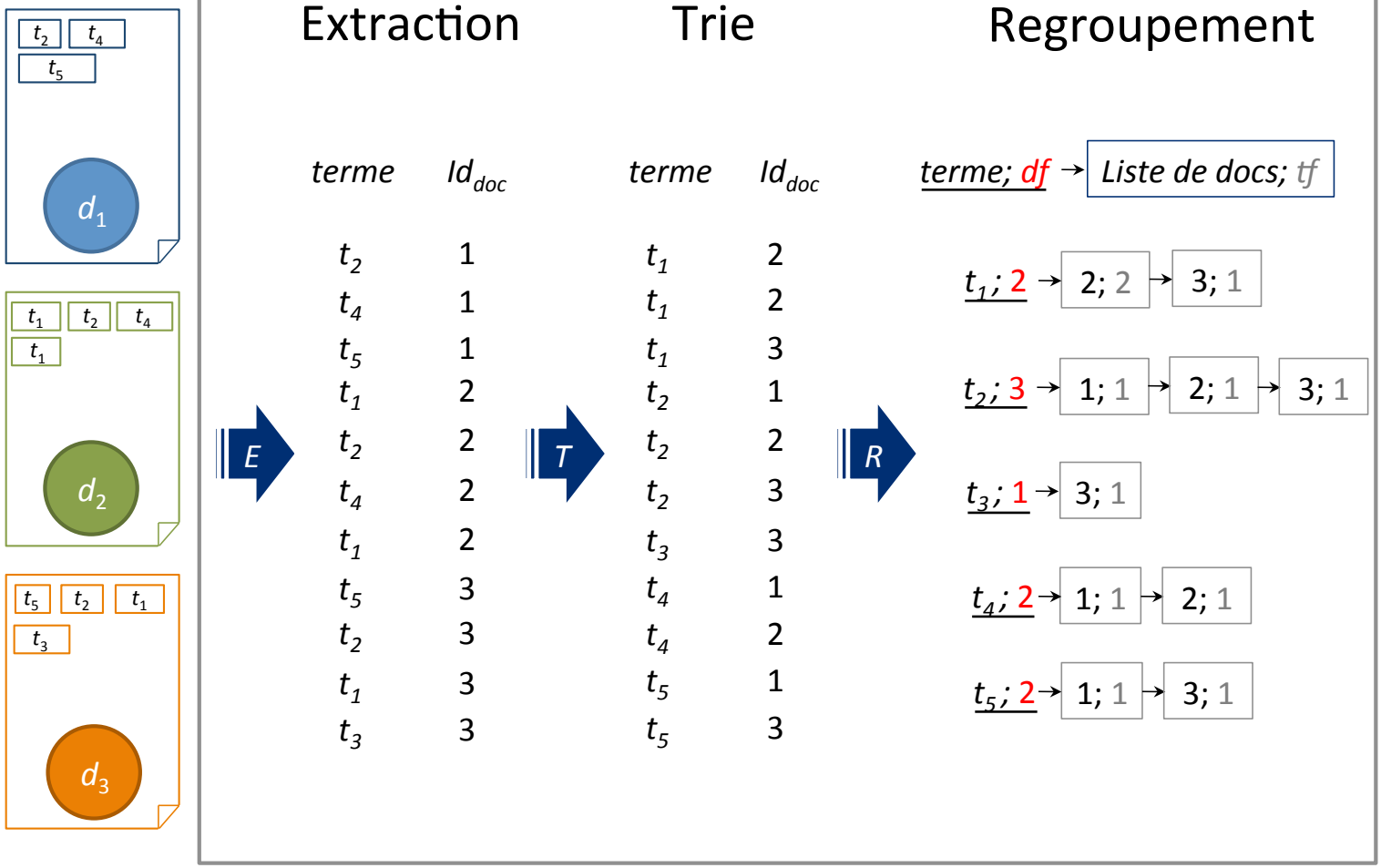
document d {
- int l — (long. du doc. (types))
- TabTermes int[l] — (indices des termes)
- TabPoids float[l] — (poids des termes)
- ...

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.
- Les deux types de structure de données les plus utilisées pour la construction de l'index inversé sont des *tables de hachage* ou des *arbres*.
- Pour un meilleur rendement, les systèmes de recherche mettent généralement le *vocabulaire* et une partie des *listes des identifiants de documents* dans la mémoire vive, les autres informations étant stockées sur disque.

Construction de l'index inversé : cas statique



Construction de l'index inversé : cas statique

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 ?

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. Tant que (tous les docs n'ont pas été traités)
3. faire
4. $\quad n \leftarrow n + 1$
5. $\quad$ block $\leftarrow$ ParseBlock()
6. $\quad$ BSBI-Invert(block)
7. $\quad$ WriteBlockToDisk(block,f_n)
8. Fin tant que
9. MergeBlocks(f_1, ..., f_n;f_{merged})

L'algorithme BSBI (2)

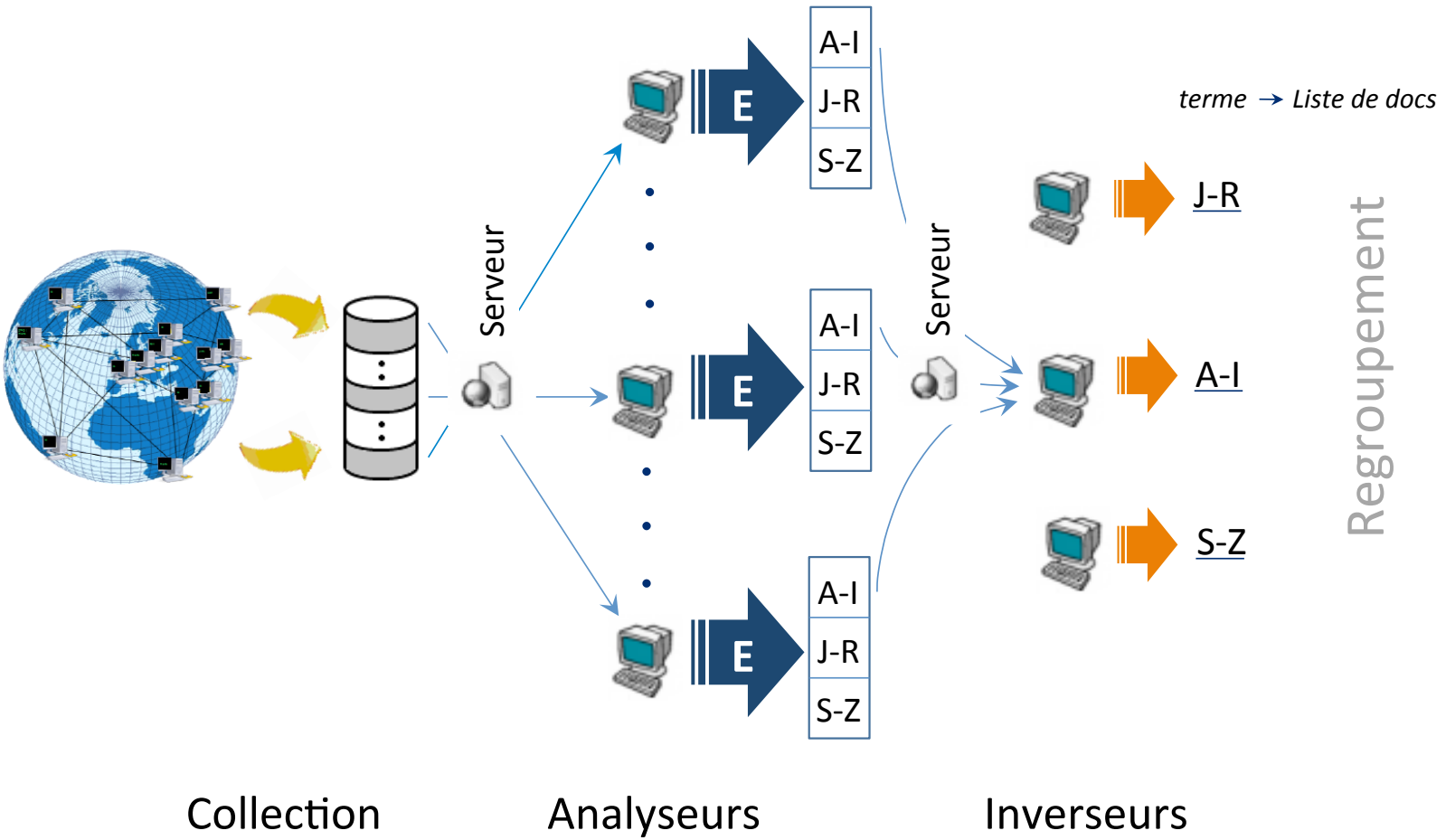
L'inversion, dans BSBI, consiste en un tri sur les identifiants de termes (premiére clé) et les identifiants de documents (deuxiéme clé). Le résultat est donc un fichier inverse pour le bloc lu. Complexité en $O(T \log T)$ où T est le nombre de paires terme-document (mais étapes de lecture et de fusion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

Exemple

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

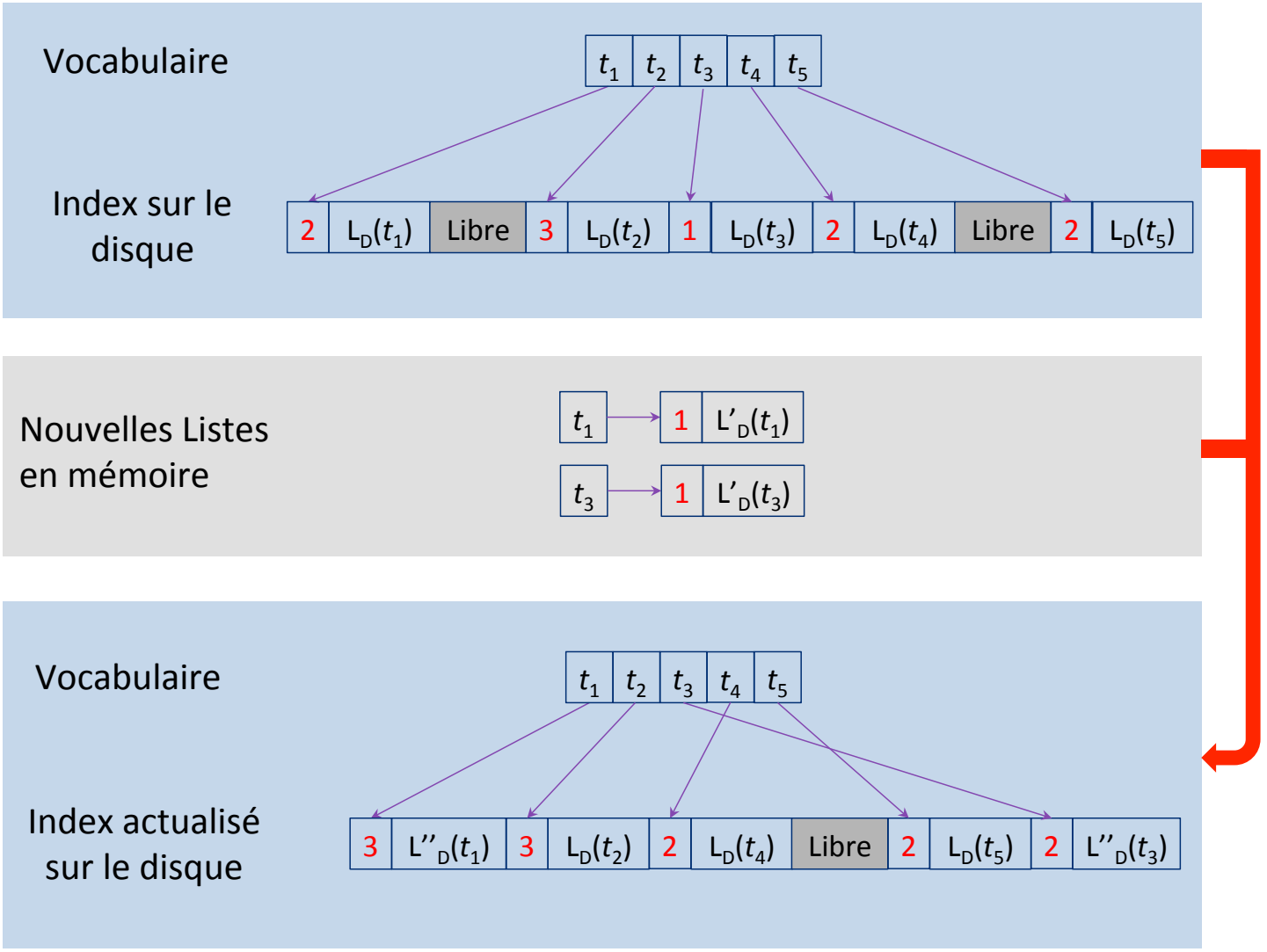
$t_1 : d_1, d_2$	$t_1 : d_5$	$t_2 : d_8$
$t_2 : d_2, d_3$	$t_2 : d_5, d_6$	$t_4 : d_8$
$t_3 : d_4$	$t_3 : d_7$	$t_5 : d_8$

Index inversé: cas distribué



Index inversé: cas dynamique

La technique de mise à jour directe sur place



Index inversé: cas dynamique

La technique de fusion