Institut Supérieur d'Informatique et des Mathématiques de Monastir

TECHNIQUES D'INDEXATION ET RECHERCHE MULTIMÉDIA

IMEN CHEBBI

Chapitre 2

1.Indexation

Recherche d’information (RI) :

Ensemble des méthodes et techniques pour l’acquisition, l’organisation, le stockage, la recherche et la **sélection d’information pertinente pour un utilisateur**

Un Système de Recherche d’Information (SRI)

Un système de recherche d'information (RI) est un système qui permet de retrouver les documents pertinents à une requête d'utilisateur, à partir d'une base de documents volumineuse.

Trois notions clés: **documents**, **requête**, **pertinence**.

Requête : exprime le besoin d'information d'un utilisateur

Document : toute unité qui peut constituer une réponse à une requête,
Un document peut être un texte, un morceau de texte, une page Web, une image, une bande vidéo, etc,

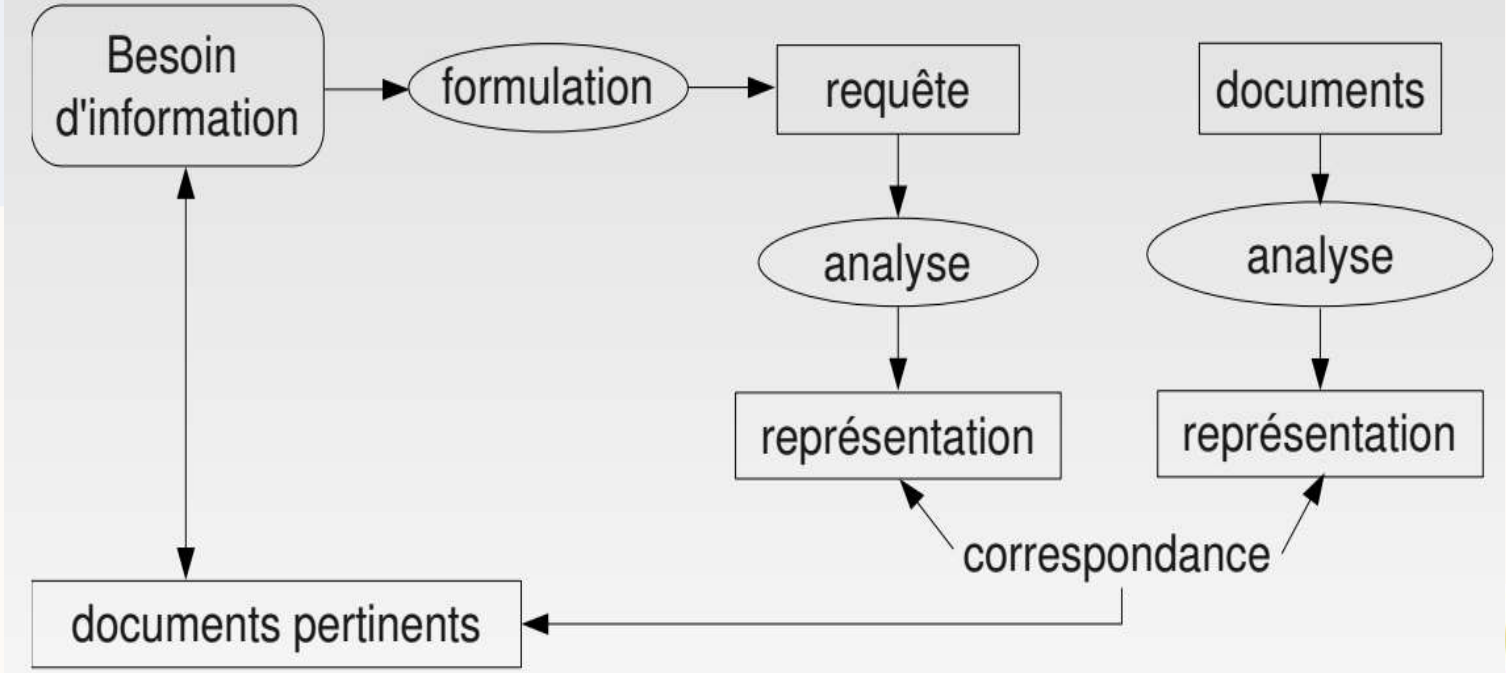
Base de documents : ensemble des documents disponibles

Pertinence : De façon générale, dans document pertinent, l'utilisateur doit pouvoir trouver les informations dont il a besoin. Sur cette notion le système doit juger si un document doit être donné à l'utilisateur comme réponse ou non

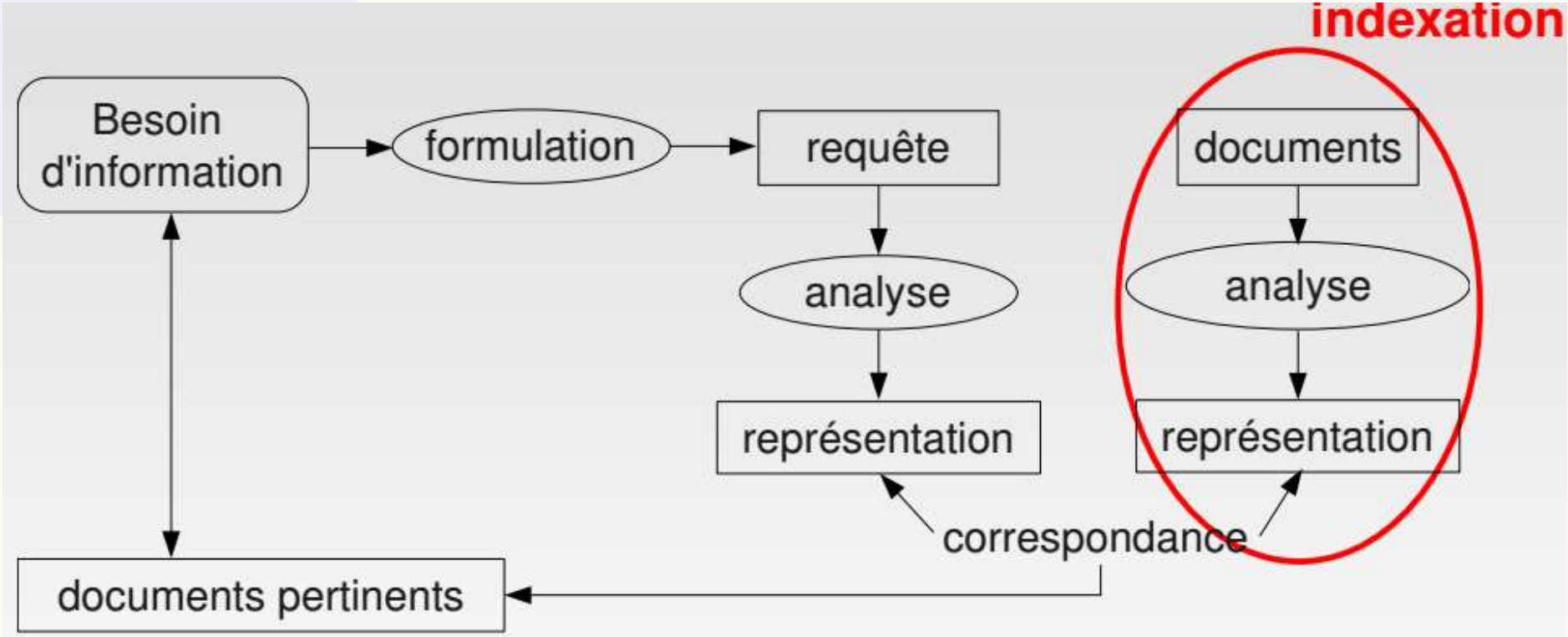
EXEMPLES D'APPLICATIONS

- Outils de recherche dans les mails, dans les fichiers, ...
- Systèmes de RI documentaires,
- Systèmes de RI pour les bases de documents d'une entreprise,
- Systèmes de RI sur le Web tels que google, bing ,,,etc.

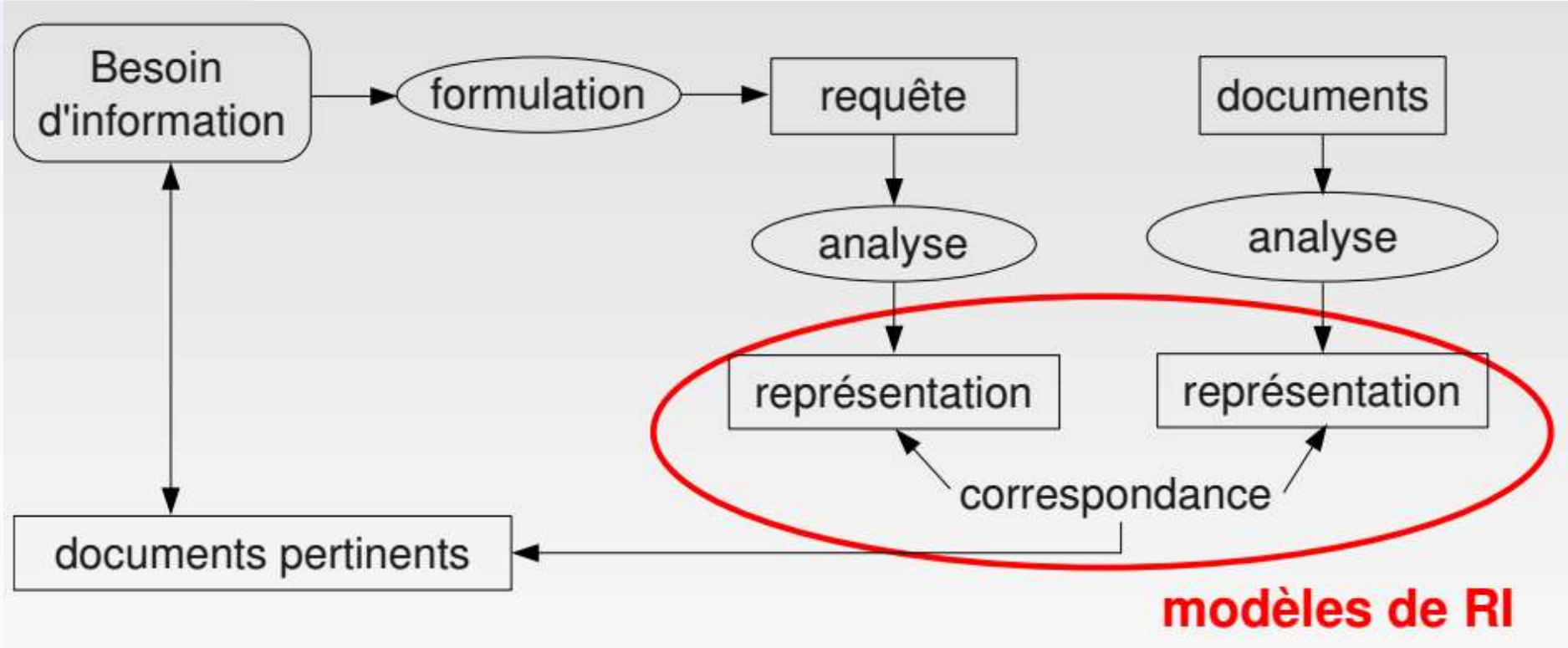
APPROCHE CLASSIQUE DE LA RI



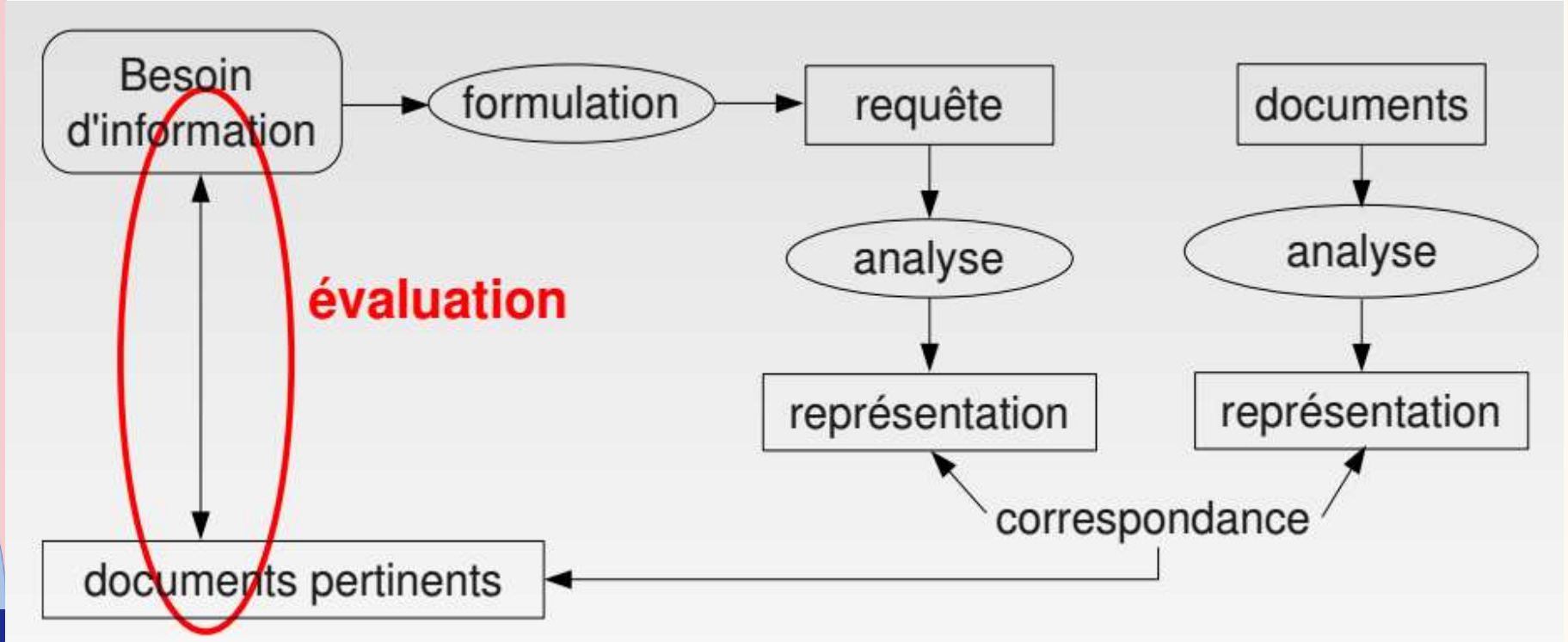
APPROCHE CLASSIQUE DE LA RI



APPROCHE CLASSIQUE DE LA RI



APPROCHE CLASSIQUE DE LA RI



INDEXATION

base
de
données

INDEXATION - POURQUOI UTILISER LES INDEX ?

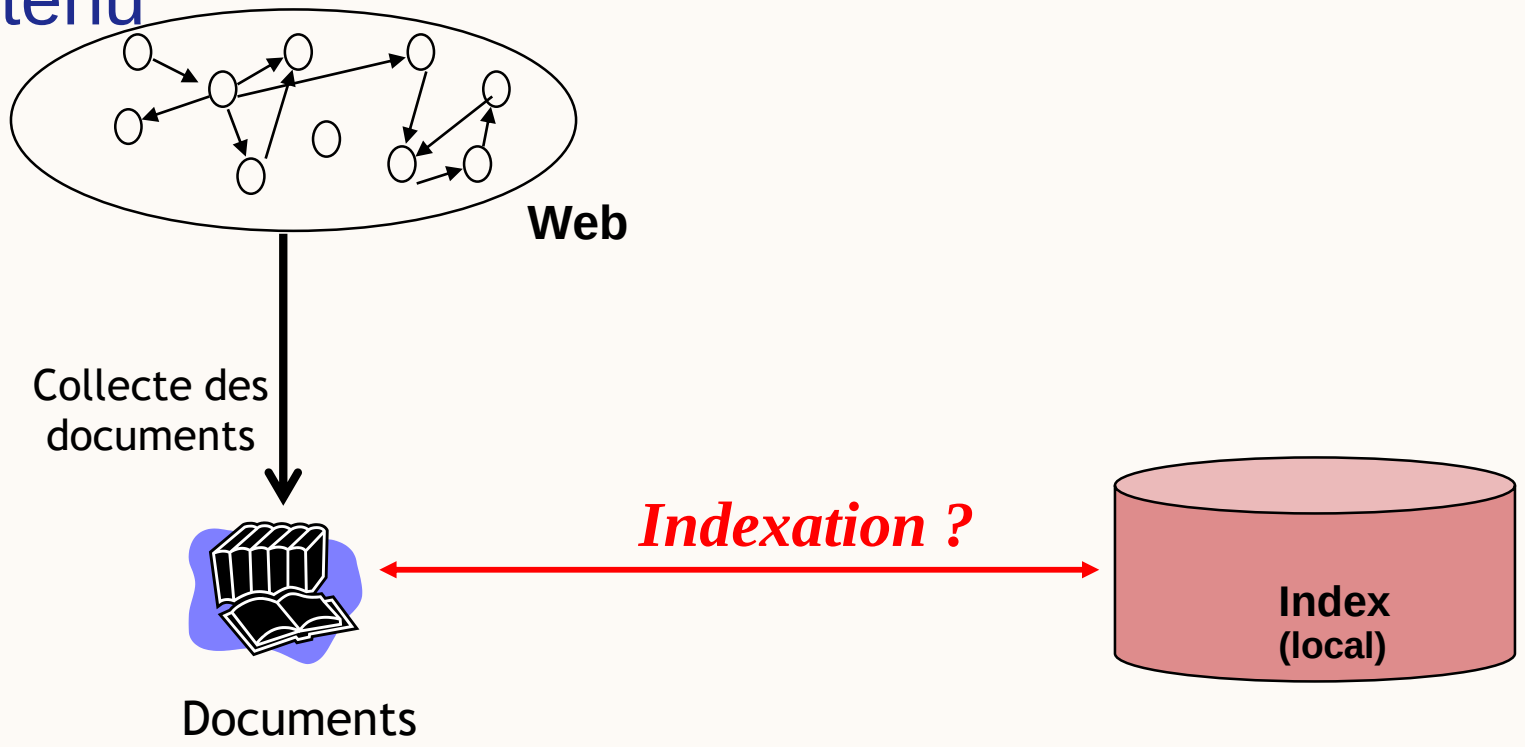
- Imaginez un moteur de recherche qui ne dispose pas d'une base d'index

- Pour chaque requête, il doit
  - **accéder au Web (faire un tour complet)**
  - **analyser les documents un par un**
  - **juger l'importance de chaque document par rapport à la requête en question**
  - **« fabriquer" la réponse en fonction des pertinences des documents**
  - **afficher le résultat**

=> une base d'index est indispensable

INDEXATION

- Analyse du document et interprétation de son contenu



INDEXATION

- Un index contient une "interprétation" du document au lieu du document entier

- Il contient
  - les termes représentatifs d'un document
  - les poids (l'importance) des termes dans chaque document

- Chaque moteur possède un index inverse
  - transformation de

    "quels mots apparaissent dans la page ?" en **"dans quelles pages (URL) apparaît le mot X?"**

INDEXATION

- Peut être
  - Manuelle (expert en indexation)
  - Automatique (ordinateur)
  - Semi-automatique (combinaison des deux)
- Basée sur
  - Un langage contrôlé (lexique/thesaurus/ontologie/réseau sémantique)
  - Un langage libre (éléments pris directement des documents)

INDEXATION MANUELLE

- Choix des mots effectué par des indexeurs
- Basée sur un vocabulaire contrôlé
- Approche utilisée souvent dans les bibliothèques, les centres de documentation
- Dépend du savoir faire de l’indexeur

INDEXATION MANUELLE: AVANTAGE DU VOCABULAIRE CONTRÔLE

- Permet la recherche par concepts (par sujets, par thèmes), plus intéressante que la recherche par mots simples.
- Permet la classification (regroupement) de documents (par sujets, par thème).
- Fournit une terminologie standard pour indexer et rechercher les documents

INDEXATION MANUELLE: INCOVÉNIENT DU VOCABULAIRE CONTRÔLE

Indexation très coûteuse

– Pour construire le vocabulaire
– Pour affecter les concepts (termes) aux documents (**imaginer cette opération sur le web**)

Difficile à maintenir

– La terminologie évolue, plusieurs termes sont rajoutés tous les jours

Processus humain donc subjectif

– Des termes différents peuvent être affectés à un même document par des indexeurs différents

Les utilisateurs ne connaissent pas forcément le vocabulaire utilisé

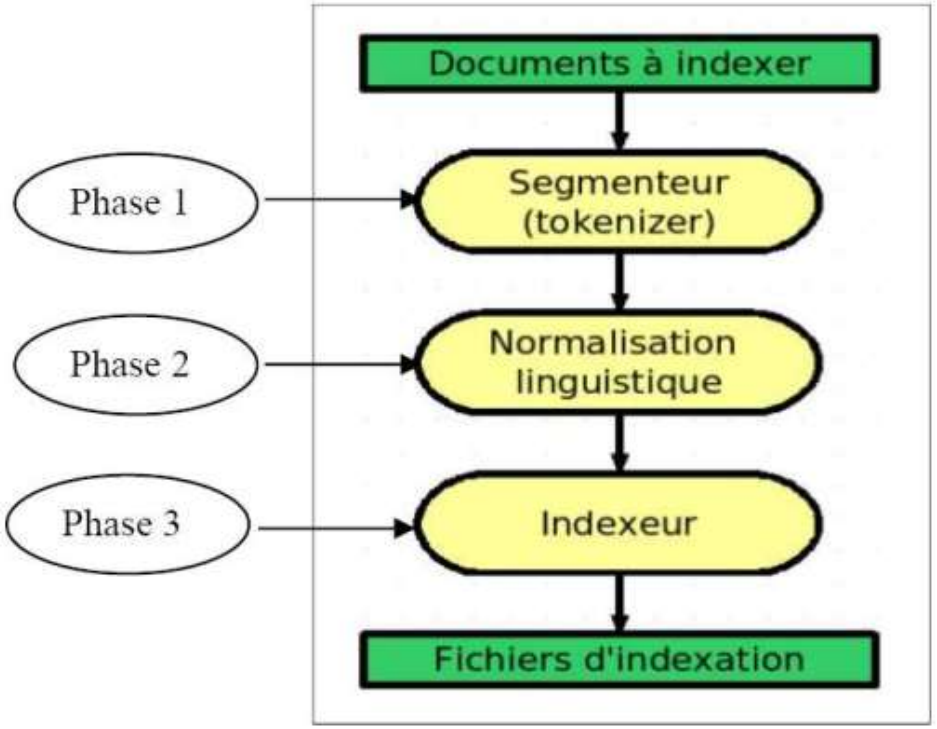
par les indexeurs

INDEXATION AUTOMATIQUE

- **Approches basées sur**
  - Statistique (distribution des mots) et/ou TALN (compréhension du texte)
- **Approche repose sur des hypothèses simples:**
  - Redondance d’un mot marque son importance
  - Cooccurrence des mots marque le sujet d’un document

INDEXATION AUTOMATIQUE: PROCESSUS

- Le processus de l'indexation se constitue de :



INDEXATION AUTOMATIQUE
ETAPE 1 : EXTRACTION DES MOTS

- Extraire les termes (tockenization)
  – terme = suite de caractères séparés par (blanc ou signe de ponctuation, caractères spéciaux,…), Nombres
- Ce sont les index utilisés lors de la recherche
- Dépend de la langue

INDEXATION AUTOMATIQUE
ETAPE 2 : NORMALISATION

- Cette phase peut contenir plusieurs étapes.
- Les étapes les plus importantes et les plus utilisées :
  1. *Elimination des mots vides*
  2. *La racinisation* (« stemming » en anglais)
  3. *La lemmatisation*
  4. *Extraction des mots composés*
  5. *Extraction des entités nommées,*

NORMALISATION: ELIMINATION DES MOTS VIDES

- Les mots vides (article, proposition, conjonction, etc.) sont des mots non significatifs dans un document, car ils ne traitent pas le sujet du document.
- On distingue deux techniques pour éliminer les mots vides :
  - L'utilisation d'une liste préétablie de mots vides (*stop-words*),
  - L'élimination des mots ayant une fréquence qui dépasse un certain seuil dans la collection.
- L'élimination des mots vides réduit la taille de l'index, ce qui améliore le temps de réponse du système.

NORMALISATION: STEMMING

- La normalisation consiste à représenter les différentes variantes d'un terme par un format unique appelé lemme ou racine. Ce qui a pour effet de réduire la taille de l'index.

- Plusieurs stratégies de normalisation sont utilisées :
  -la table de correspondance,
  -l'élimination des affixes **(l'algorithme de Porter),**
  -la troncature,
  -l'utilisation des N-grammes.

NORMALISATION: STEMMING

- Exemple:

  - économie, économiquement, économiste, ➔ économ
  - pour l'anglais : retrieve, retrieving, retrieval, retrieved, retrieves ➔ retriev
- L'inconvénient majeur de cette opération est qu'elle supprime dans certains cas la sémantique des termes originaux,

INDEXATION AUTOMATIQUE: INDEX INVERSÉ

- Une fois les documents indexés :
  – chaque document aura donc un descripteur (une liste de mots souvent simples): à Sac de mots (Bag of Words)
  – Ces termes sont ensuite stockés dans une structure appelée fichier inverse.

INDEXATION AUTOMATIQUE: INDEX INVERSÉ

- Dans sa forme la plus simple, l'index inversé d'une collection de documents est essentiellement une structure de données qui relie chaque terme distinct à une liste de tous les documents qui le contiennent.

INDEX INVERSÉ

- Un index inversé se compose de deux parties:

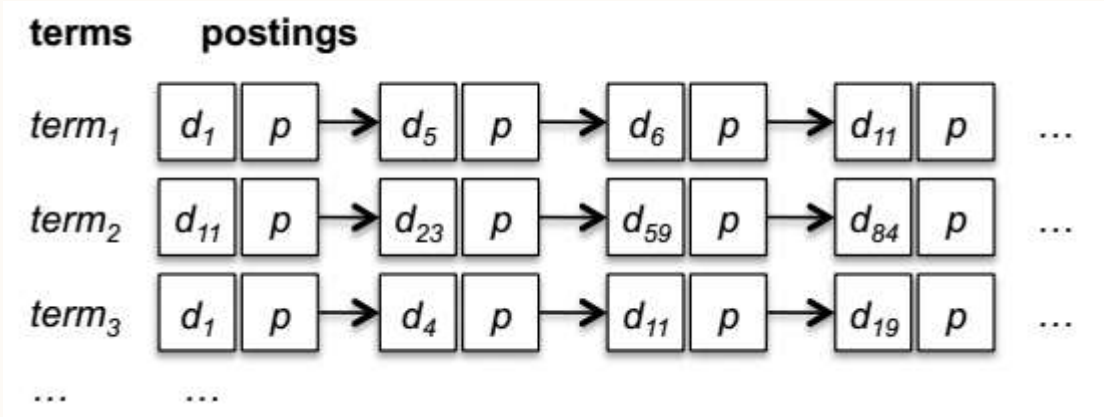
1. Un vocabulaire V, contenant tous les termes distincts de l'ensemble de documents, et
2. pour chaque terme distinct, une liste inversée de publications.

INDEX INVERSÉ

- Chaque enregistrement stocke l'ID (désigné par id_j) du document d_j qui contient le terme t_i et d'autres informations sur ce terme dans ce document.
- Selon le besoin de l'algorithme de recherche ou de classement, différentes informations peuvent être incluses.

INDEX INVERSÉ

- Pour chaque terme, nous avons une liste qui enregistre dans quels documents le terme se produit.
- Chaque terme de la liste est appelé classiquement **Posting** (publication).
- Un Posting est un tuple de la forme(t_i,d_j), où t_i est un identificateur de terme et d_j est un identifiant de document.
- La liste est appelée liste de posting (ou liste inversée)



INDEX INVERSÉ

- Les index inversés sont indépendants du modèle IR adopté (Modèle booléen, modèle d'espace vectoriel, etc.)
- Chaque Posting contient généralement:
  - L'identifiant du document lié.
  - La fréquence d'apparition du terme dans le document
  - La position du terme pour chaque document (facultatif)
    - Exprimé en nombre de mots depuis le début du document, le nombre d'octets, etc.
- Pour chaque terme est également généralement stocké la fréquence d'apparition du terme dans l’ensemble des documents.

INDEX INVERSÉ: CONSTRUCTION

- Extraire les termes de chaque document dans un fichier (1fichier par document) ou un fichier pour plusieurs documents)

Doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

INDEX INVERSÉ: CONSTRUCTION

- Trier le fichier termes-documents:

Trier le fichier par ordre alphabétique des termes et par document

Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

INDEX INVERSÉ: CONSTRUCTION

- Pour chaque terme,
  - on dispose de la liste de documents qui le contient
  - Le nombre de documents comportant ce terme

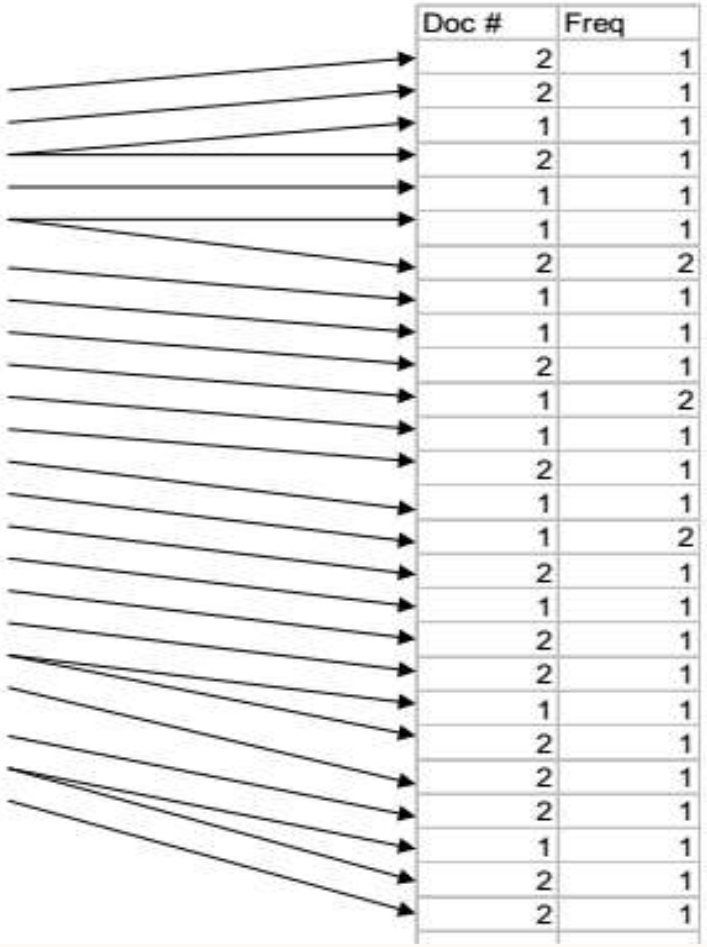
Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Term	Doc #	Freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

INDEX INVERSÉ: CONSTRUCTION

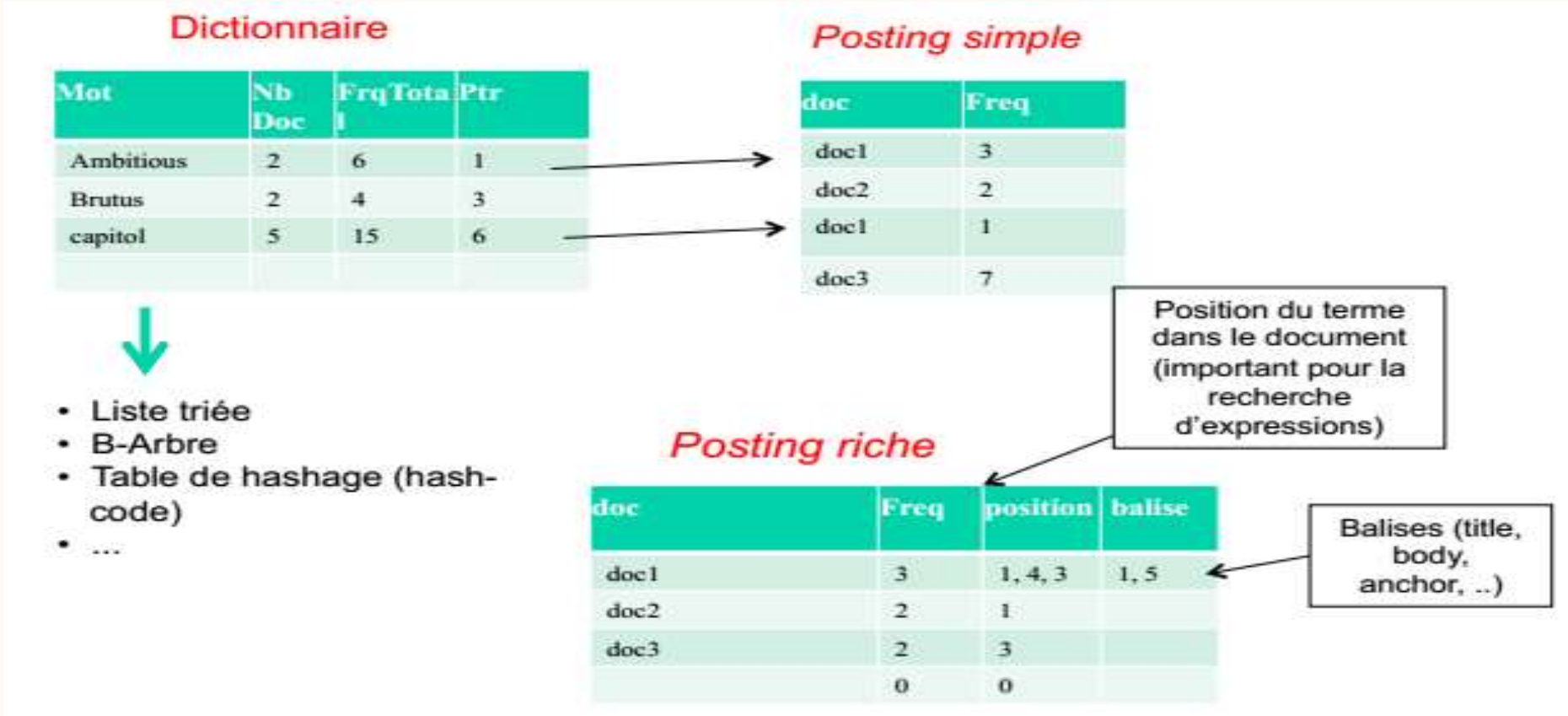
Term	Doc #	Freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

Term	N docs	Tot Freq
ambitious	1	1
be	1	1
brutus	2	2
capitol	1	1
caesar	2	3
did	1	1
enact	1	1
hath	1	1
I	1	2
i'	1	1
it	1	1
julius	1	1
killed	1	2
let	1	1
me	1	1
noble	1	1
so	1	1
the	2	2
told	1	1
you	1	1
was	2	2
with	1	1



Doc #	Freq
2	1
2	1
1	1
2	1
1	1
1	1
2	2
1	1
1	1
2	1
1	2
1	1
2	1
1	1
1	2
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1

INDEX INVERSÉ: ORGANISATION



INDEX INVERSÉ: STOCKAGE

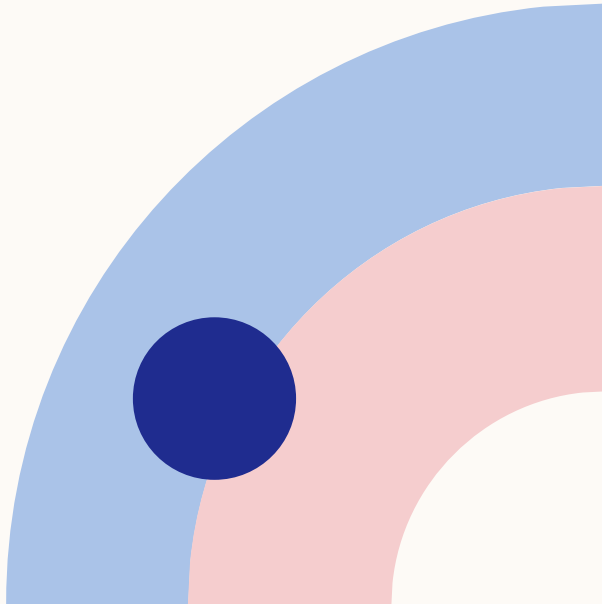
Les termes se trouvent généralement dans un certain nombre de documents:

- Les index inversés réduisent les besoins de stockage de l'index,
- fournir la base pour une recherche efficace
- cette structure d'index inversé est essentiellement sans rivaux comme la structure la plus efficace pour supporter la recherche de texte.

INDEX INVERSÉ: STOCKAGE

Les listes liées généralement préférées aux tableaux:

- Allocation dynamique de l'espace
- L'insertion de termes dans les documents est facile
- Frais généraux des pointeurs
- Espace requis

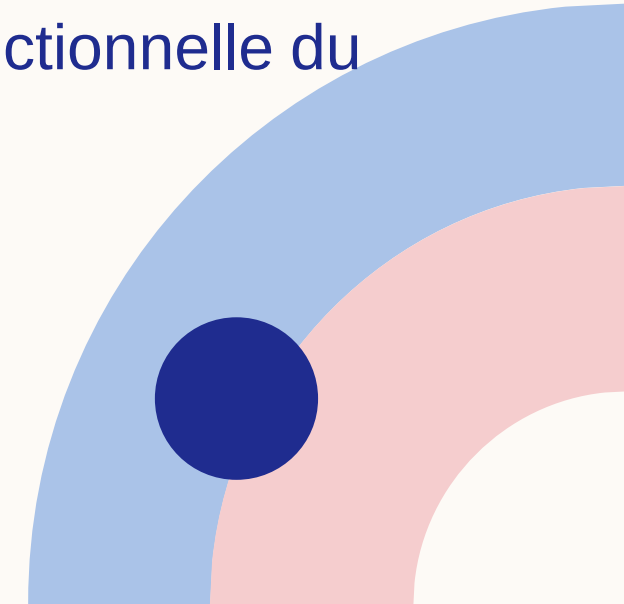


INDEXATION DISTRIBUÉE

Pour de très larges collections (Web).

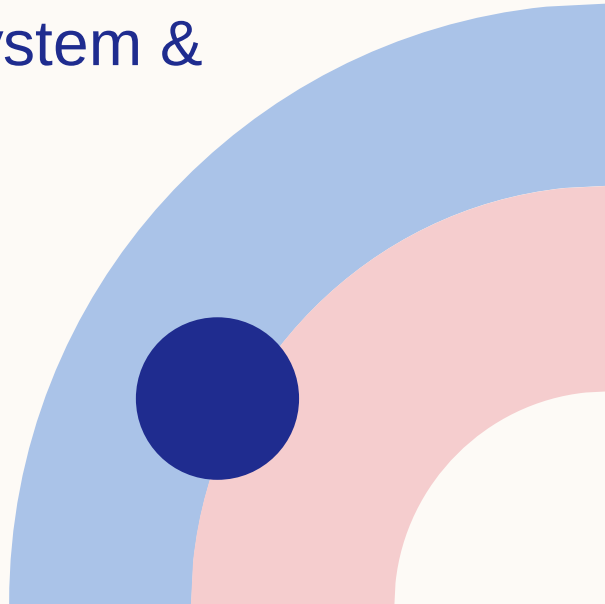
Un serveur principal dirige le tout

- Il divise la tâche d'indexation en un ensemble de tâches parallèles
- Il assigne chaque tâche à une machine libre et fonctionnelle du réseau



INDEXATION DISTRIBUÉE

- Les moteurs de recherche utilisent une architecture semblable
  - un système de fichiers distribué
  - un système de contrôle de tâches (job scheduler : quel programme est exécuté sur quelle machine à quel moment):
  - Architecture initiale proposée par Google (Google File System & Map Reduce)
  - Implémentation libre développée dans le projet Hadoop



MAPREDUCE

Principe :

– Tous les algorithmes sont écrits sous la forme de deux fonctions :

1. Une fonction ***map*** qui réalise un traitement des données
2. Une fonction ***reduce*** qui fusionne les résultats intermédiaires produits par *map*

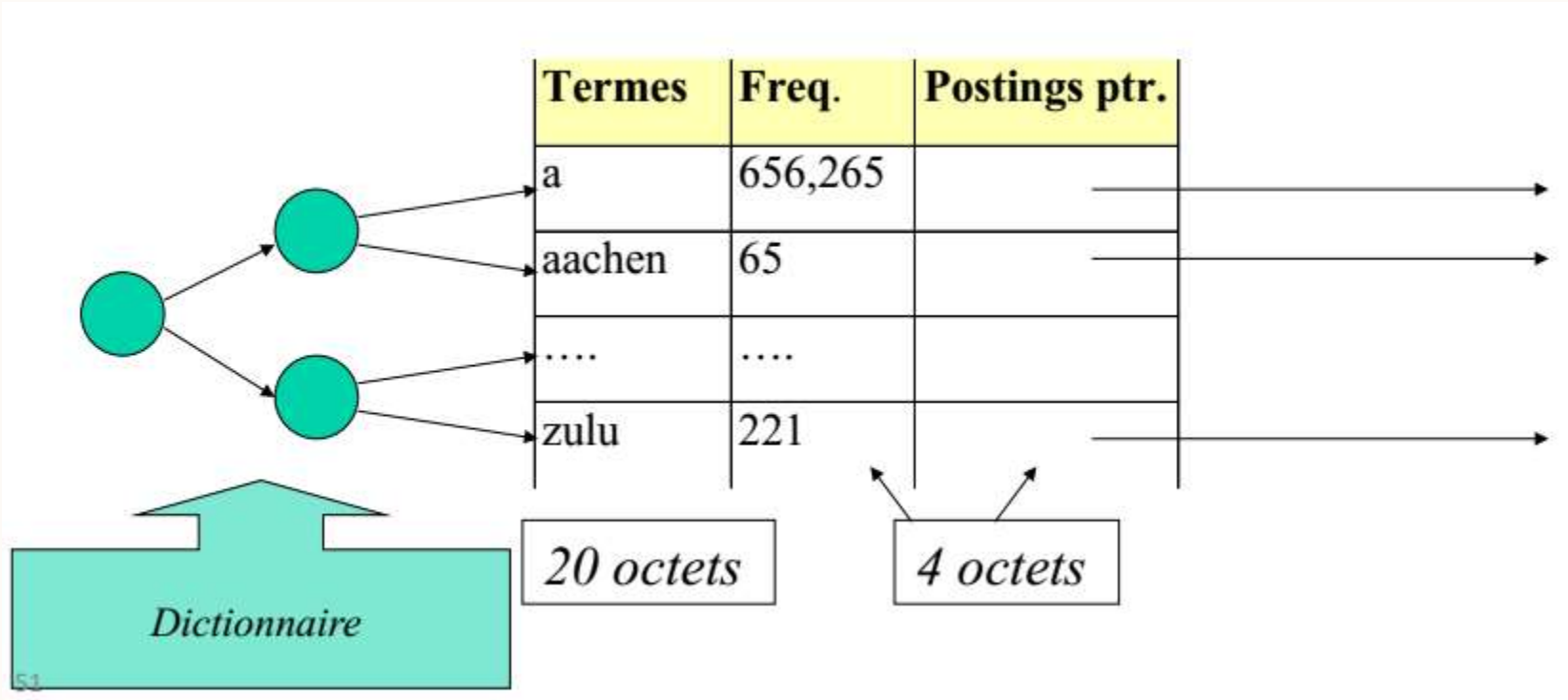
Interêt :

– Les tâches *map* sont exécutées en parallèle sur les machines sur lesquelles sont stockées les données.

INDEXE INVERSÉ: COMPRESSION

COMPRESSION DU DICTIONNAIRE

- Taille du dictionnaire:
  - Tableau de taille fixe
    - ~400,000 termes; 28 octets/terme = 11.2 MO.



COMPRESSION DU DICTIONNAIRE

- Beaucoup d’espace perdu, les mots à une lettre (a, à, ..)

occupe le même espace que des mots longs

– Il y a des mots qui ne passent pas « anticonstitutionnellement »
« *supercalifragilis,cexpialidocious* » ou « *hydrochlorofluorocarbons* ».

COMPRESSION DU DICTIONNAIRE

- Taille moyenne des mots (en anglais), elle est autour de ~8 caractères

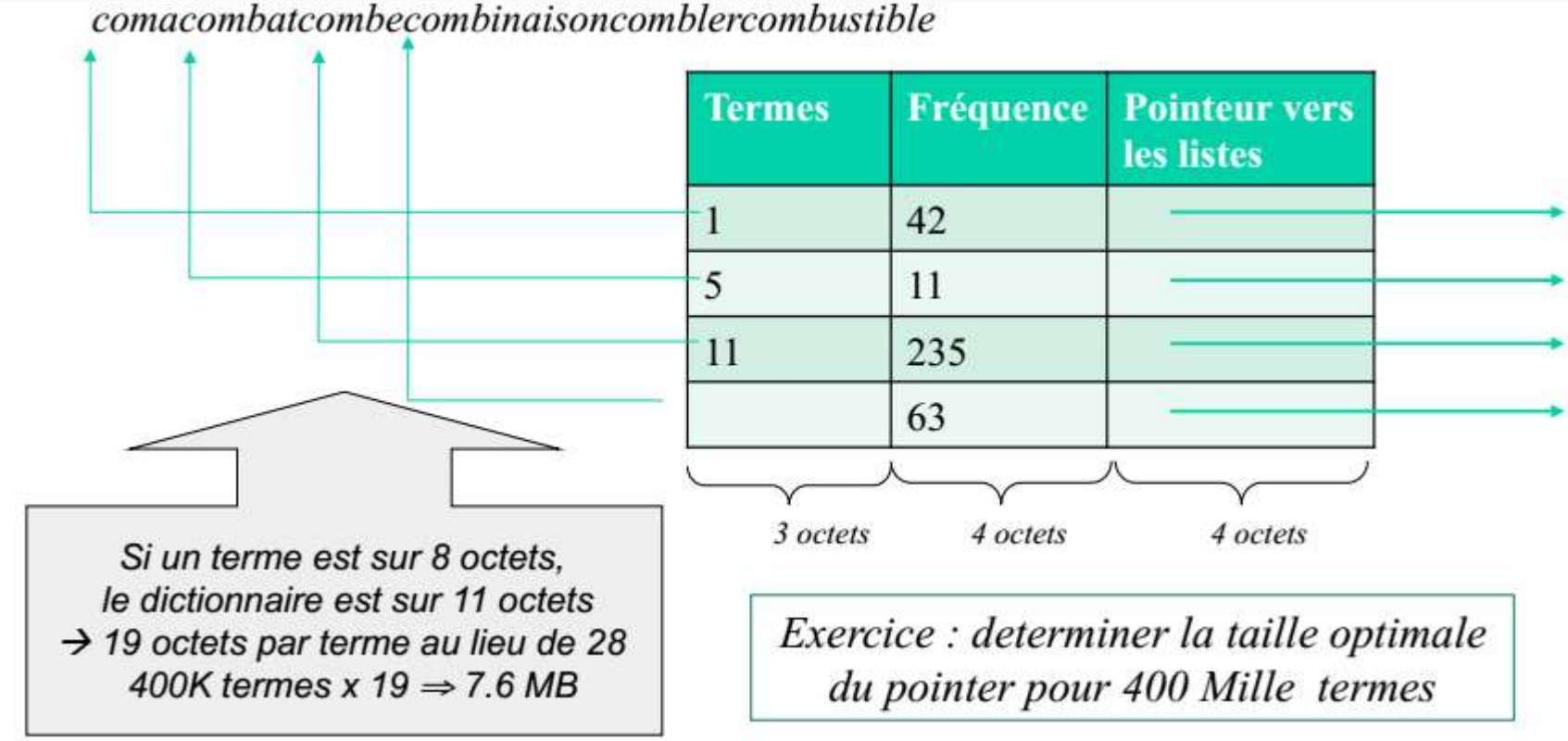
– Comment peut-on exploiter ce nombre

(~8 caractères par terme)?

COMPRESSION DU DICTIONNAIRE

Stocker le dictionnaire comme un (long) string de caractères

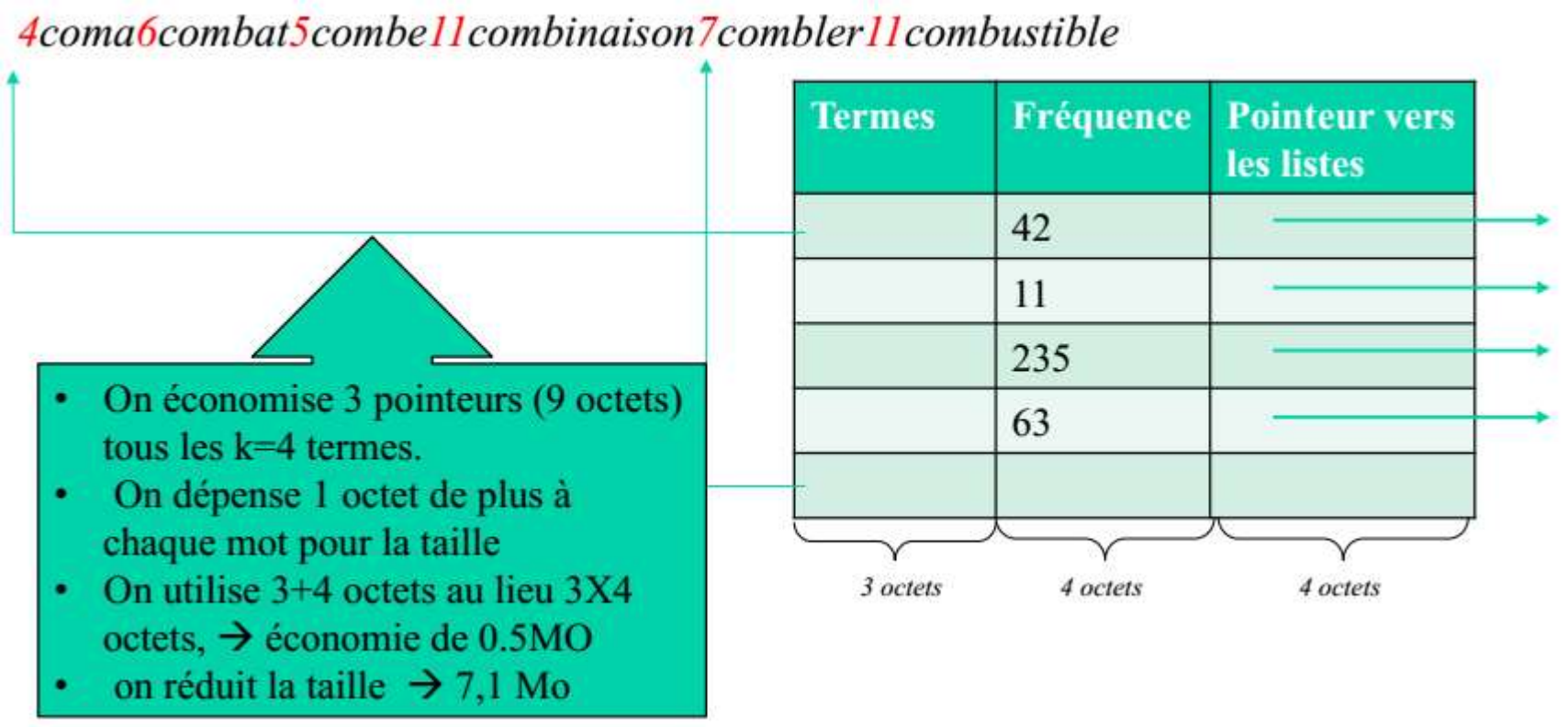
- Pointeur vers le terme suivant donne la fin du terme courant



COMPRESSION DE LA LISTE DES TERMES

Stocker les pointeurs à chaque k termes (Exemple : k=4).

Besoin de rajouter un octet pour stocker la taille du terme



COMPRESSION DU *POSTING*

Le fichier *Posting* est au moins 10 fois plus volumineux que le dictionnaire

– Le *Posting* est formé de DocId(s) (numéro de document)

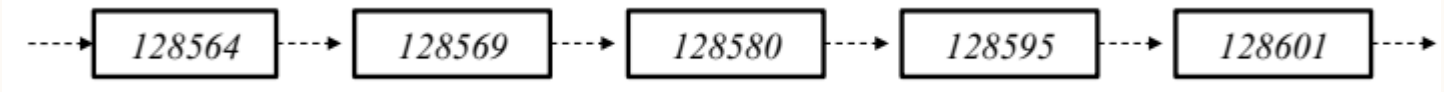
– Au mieux, pour 1 M de documents, sur $\log_2$ 1 000 000 ≈ 20 bits

COMPRESSION DU *POSTING*

Peu de termes **fréquents**, **beaucoup** de termes **rares**:

– « ***arachnocentric*** » apparaît peut-être une fois dans toute la collection à donc pour notre collection d'un million de documents, 20 bits devraient suffire.

– « **the** » apparaît probablement dans tous les documents, donc potentiellement 20bitsX1M=20M de bits pour stocker la liste (c'est trop!!!)



COMPRESSION DU *POSTING*

Les docid(s) du *posting* sont stockés par ordre croissant

– ***computer***: 33,47,154,159,202 …

Proposition:

Stocker l'écart (intervalle) entre les docid(s) au lieu des docids .

– ***computer***: 33,14,107,5,43 …

L'espoir est de pouvoir stocker les écarts (intervalles) dans moins de 20 bits

(moins de bits que si l'on gardait les docIds)

Exemple:

	encoding	postings list								
THE	docIDs	...		283042		283043		283044		283045 ...
	gaps				1		1		1	...
COMPUTER	docIDs	...		283047		283154		283159		283202 ...
	gaps				107		5		43	...
ARACHNOCENTRIC	docIDs	252000		500100						
	gaps	252000	248100							

COMPRESSION DU *POSTING*

But:

- Pour ***arachnocentric***, on ulilise ~20 bits/écart .
- Pour ***the***, on peut utiliser ~1 bit/écart.

à Pour une valeur d'écart *l*, on veut utiliser aussi peu de bits possible (l'entier au-dessus de log2 *l*).

En pratique, on arrondit à l'octet supérieur

➔ vers l' Encodage ***variable***

COMPRESSION DU *POSTING:* ENCODAGE *VARIABLE*

- On consacre 7 bits d'un octet à représenter le nombre (l'écart), et le dernier est le bit de continuation *c*.
  - Si *l* ≤ *127*, 7 bits suffisent alors *c* = *1. (c-à-d* le nombre se termine à cet octet*)*
  - Sinon, *c* = *0* et on continue sur l'octet suivant.

COMPRESSION DU *POSTING:* ENCODAGE *VARIABLE*

- Exemple:

docIDs	824	829	215406
gaps		5	214577
VB code	00000110 10111000	10000101	00001101 00001100 10110001

Postings stockés comme concaténattion d'octets

000001101011100010000101000011010000110010110001

Pour de petits écarts (5), VB Utilise tout l'octet.

COMPRESSION DU *POSTING:* ENCODAGE *VARIABLE*

docIDs	824	829	215406
gaps		5	214577
VB code	00000110 10111000	10000101	00001101 00001100 10110001

Postings stockés comme concaténattion d'octets
0000011010111000100001010000110100001100101100001

Pour de petits écarts (5), VB Utilise tout l'octet.

Au lieu de stocker les docIDs **directement**, on stocke les **écarts (gaps)** entre deux docIDs consécutifs :

- gap_1 = 824 (souvent stocké tel quel)
- gap_2 = 829 − 824 = **5**
- gap_3 = 215406 − 829 = **214577**

☞ Les gaps sont souvent **beaucoup plus petits** → meilleure compression.

COMPRESSION DU *POSTING:* ENCODAGE *VARIABLE*

docIDs	824	829	215406
gaps		5	214577
VB code	00000110 10111000	10000101	00001101 00001100 10110001

Postings stockés comme concaténattion d'octets
0000011010111000100001010000110100001100101100001

Pour de petits écarts (5), VB Utilise tout l'octet.

Le **Variable Byte encoding** (VB) est une méthode pour
coder des entiers avec un **nombre variable d'octets**.

Principe :

- Chaque octet = **8 bits**
- **Le bit de poids fort (MSB)** indique :
  - 0 → il y a encore un octet après
  - 1 → **dernier octet du nombre**

0xxxxxxx → continuation
1xxxxxxx → fin du nombre

COMPRESSION DU *POSTING:* ENCODAGE *VARIABLE*

docIDs	824	829	215406
gaps		5	214577
VB code	00000110 10111000	10000101	00001101 00001100 10110001

Postings stockés comme concaténattion d'octets
0000011010111000100001010000110100001100101100001

Pour de petits écarts (5), VB Utilise tout l'octet.

Exemple clé du tableau : le gap = 5

Regardons la colonne **829** :
- gap = **5**
- 5 est **très petit**
- Il tient dans **un seul octet**

Codage VB de 5 : 10000101

✓ 1 → dernier octet
✓ 0000101 → valeur 5

« Pour de petits écarts (5), VB utilise tout l'octet »

COMPRESSION DU *POSTING:* ENCODAGE *VARIABLE*

docIDs	824	829	215406
gaps		5	214577
VB code	00000110 10111000	10000101	00001101 00001100 10110001

Postings stockés comme concaténattion d'octets
0000011010111000100001010000110100001100101100 01

Pour de petits écarts (5), VB Utilise tout l'octet.

Cas des grands gaps (ex : 214577)
Le gap **214577** est trop grand pour un seul octet.
☞ Il est découpé en **plusieurs octets VB** :

Exemple illustré :
00001101
00001100
10110001

•Les deux premiers commencent par 0 → continuation
•Le dernier commence par 1 → fin du nombre

COMPRESSION DU *POSTING:* ENCODAGE *VARIABLE*

docIDs	824	829	215406
gaps		5	214577
VB code	00000110 10111000	10000101	00001101 00001100 10110001

Postings stockés comme concaténattion d'octets
000001101011100010000101000011010000110010110001

Pour de petits écarts (5), VB Utilise tout l'octet.

Pourquoi "concaténation d'octets" ?

Tous les octets VB sont **collés les uns à la suite des autres** en mémoire ou dans un fichier :

00000110 10111000 10000101 00001101 00001100
10110001

☞ **Sans séparateurs**
☞ Le décodeur sait s'arrêter quand il voit un bit 1

COMPRESSION DU *POSTING:* ENCODAGE *VARIABLE*

Élément	Rôle
docIDs	Documents contenant le terme
gaps	Différences entre docIDs
VB	Compression des gaps
Petit gap	1 octet
Grand gap	Plusieurs octets
Bit MSB	Marque la fin du nombre

COMPRESSION DU *POSTING:* *AUTRES* ENCODAGE *VARIABLE*

- Au lieu des octets, nous pouvons également utiliser une "unité d'alignement" différente: 32 bits (mots), 16 bits, 4 bits etc.
- L'alignement des octets variables gaspille de l'espace si vous avez beaucoup de petits écarts
- les (4 bits) font mieux dans de tels cas.

COMPRESSION DU *POSTING:* CODE GAMMA

- Peut mieux compresser avec des codes au niveau du bit.
  - Le code Gamma est le plus connu de ceux-ci.
- Représenter un écart G en tant que paire de longueur et de décalage,
- Le décalage est G en binaire sans le bit de poids fort.
  - Par exemple 13 → 1101 → 101
- La longueur est la longueur du décalage
  - Pour 13 (décalage 101), ceci est 3.
- Coder la longueur en code unaire: 1110.
  Le code gamma de 13 est la concaténation de la longueur et du décalage: 1110101

CODE GAMMA: EXEMPLES

number	length	offset	γ-code
0			none
1	0		0
2	10	0	10,0
3	10	1	10,1
4	110	00	110,00
9	1110	001	1110,001
13	1110	101	1110,101
24	11110	1000	11110,1000
511	111111110	11111111	111111110,11111111
1025	11111111110	0000000001	11111111110,0000000001

CODE GAMMA: EXEMPLES

Le **code Gamma** est une méthode de **compression d'entiers positifs (≥ 1)**, très utilisée en **Information Retrieval** (index inversés, gaps).

⚠ **0 ne peut pas être codé** en gamma (c'est pour cela que le tableau indique none pour 0).

CODE GAMMA: EXEMPLES

Principe général du code Gamma

Pour coder un nombre **N** :

Étape 1 : écriture binaire

On écrit **N en binaire**.

Étape 2 : calcul de la longueur

Soit **L = nombre de bits du binaire de N**

Étape 3 : partie *length*

On code **(L − 1)** en **unaire** :

•L−1 fois 1, suivi d'un 0

👉 c'est la colonne **length**

Étape 4 : partie *offset*

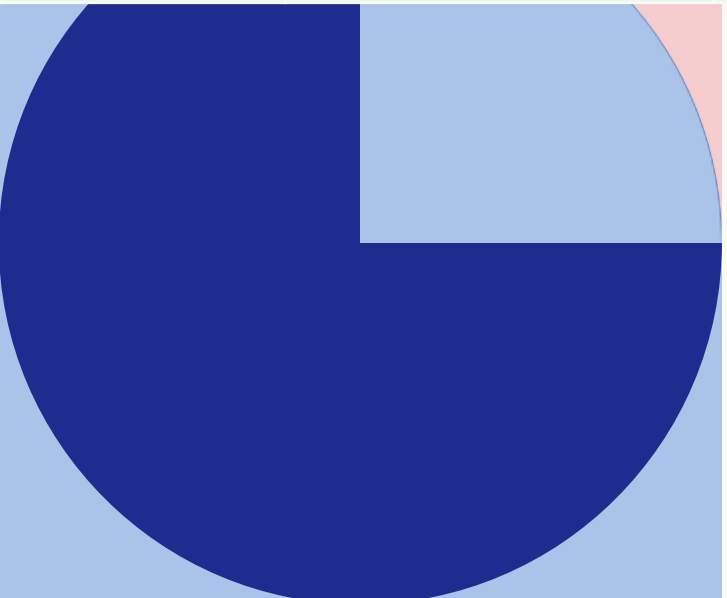
On enlève le **bit de poids fort (1)** du binaire de N

👉 le reste est l'**offset**

Étape 5 : code gamma final

γ(N) = length || offset: gamma de N est égal à *length* concaténé avec *offset*

number	length	offset	γ-code
0			none
1	0		0
2	10	0	10,0
3	10	1	10,1
4	110	00	110,00
9	1110	001	1110,001
13	1110	101	1110,101
24	11110	1000	11110,1000
511	111111110	11111111	111111110,11111111
1025	11111111110	0000000001	11111111110,0000000001



Nombre = 1
- Binaire : 1
- L = 1
- length = 0
- offset = (rien)
- γ-code = 0

✓ correspond au tableau

Nombre = 2
- Binaire : 10
- L = 2
- length = 10
- offset = 0
- γ-code = 10,0

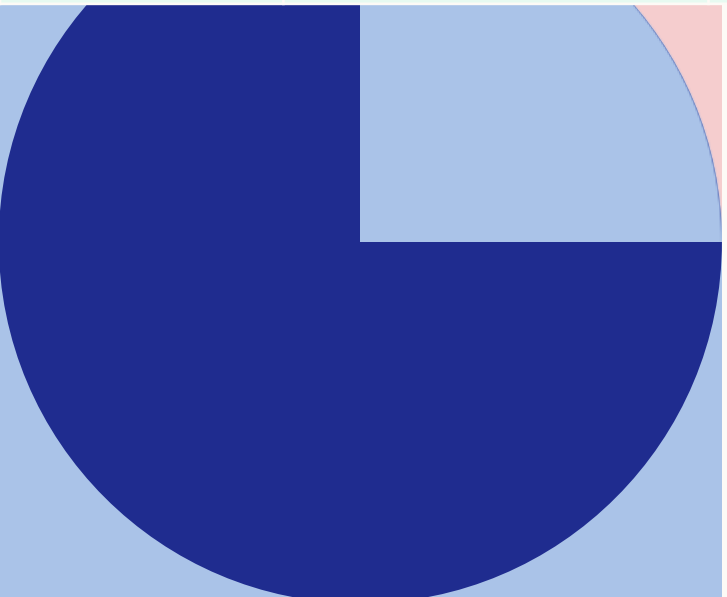
Nombre = 3
- Binaire : 11
- L = 2
- length = 10
- offset = 1
- γ-code = 10,1

Nombre = 1025
- Binaire : 10000000001
- L = 11
- length = 11111111110
- offset = 0000000001
- γ-code = 11111111110,0000000001

number	length	offset	γ-code
0			none
1	0		0
2	10	0	10,0
3	10	1	10,1
4	110	00	110,00
9	1110	001	1110,001
13	1110	101	1110,101
24	11110	1000	11110,1000
511	111111110	11111111	111111110,11111111
1025	11111111110	0000000001	11111111110,0000000001

Nombre = 13
- Binaire : 1101
- L = 4
- length = 1110
- offset = 101
- γ-code = 1110,101

Nombre = 24
- Binaire : 11000
- L = 5
- length = 11110
- offset = 1000
- γ-code = 11110,1000



Nombre = 4
- Binaire : 100
- L = 3
- length = 110
- offset = 00
- γ-code = 110,00

Nombre = 9
- Binaire : 1001
- L = 4
- length = 1110
- offset = 001
- γ-code = 1110,001

Nombre = 511
- Binaire : 111111111
- L = 9
- length = 111111110
- offset = 11111111
- γ-code = 111111110,11111111

Pourquoi Gamma est efficace ?

✓ Très efficace pour **petits nombres**
✓ Très utilisé pour coder les **gaps** dans les listes de postings
✓ Décodage sans séparateurs (auto-délimitant)

✗ Moins efficace que VB pour les **très grands nombres**

Élément	Rôle
length	indique la taille du nombre
offset	valeur réelle sans le MSB
γ-code	length + offset
Domaine	entiers ≥ 1
Usage	gaps, index inversés

INDEXATION: RÉCAPITULATIF

- Opération FONDAMENTALE en RI,
- Elle permet la sélection des termes importants caractérisant le contenu d'un document
- Elle peut être
  - manuelle/automatique
- approche courante plutôt automatique
  - linguistique / statistique
- approche courante plutôt statistique
  - Idéalement combinaison linguistique + statistiqu

INDEXATION: RÉCAPITULATIF

- A l'issue de cette opération, chaque document sera représenté par
une liste de termes pondérés.
- Le poids est fondamental et a une grande influence dans toutes
les approches (modèles) de RI.
- L'ensemble des termes extraits de tous les documents est stocké dans une structure spécifique appelée : fichier inverse.
- Ce fichier permet de retrouver pour un terme donné tous les documents qui contiennent ce terme.

MERCI