Représentation de l'information : Indexation

Représentation de l'information

- Représentation de l'information = indexation
  - Processus permettant de construire un ensemble d'éléments « clés » permettant de caractériser le contenu d'un document / retrouver ce document en réponse à une requête

- Éléments clés
  - Information textuelle
    - mots simples : pomme
    - groupe de mots : pomme de terre
  - Image
    - Couleurs, formes

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)

Vocabulaire contrôlé

- Lexique
  - Liste de mots clés
- Liste hiérarchique
    - de concepts
    - de notations (codes)
- Thésaurus
  - Liste de mots clés + relation sémantiques entre les mots clés
- Ontologie
  - Liste concepts + relations entre les concepts

Exemple de liste hiérarchique

- MeSH (Medical Subject Headings)

(MeSH Tree Structures - 2003.htm)

A. Anatomy

B. Organisms

C. Diseases

C1. Bacterial infections

C2. Virus diseases

arbovirus infection

Encephalitis, Epidemic

C3. Parasitic diseases

Exemple de liste hiérarchique

- Yahoo Yahoo! France.htm

Guide des sites Internet - Classement thématique des sites

Actualités et médias
Journaux, Télévision, Météo...

Sports et loisirs
Foot, Tourisme, Auto/Moto, Jeux...

Commerce et économie
B2B, Shopping, Emploi, Immobilier...

Art et culture
Littérature, Cinéma, Musique, Musées...

Informatique et Internet
Internet, Logiciels, Matériel...

Divertissement
À voir, Loteries, Humour, Sorties...

Santé
Diététique, Médecine, Organismes...

Exploration géographique
Zones régionales, Pays, Europe, France...

Enseignement et formation
Primaire, Secondaire, Supérieur...

Références et annuaires
Dictionnaires, Annuaires, Bibliothèques...

Institutions et politique
Ministères, Droit, Services publics...

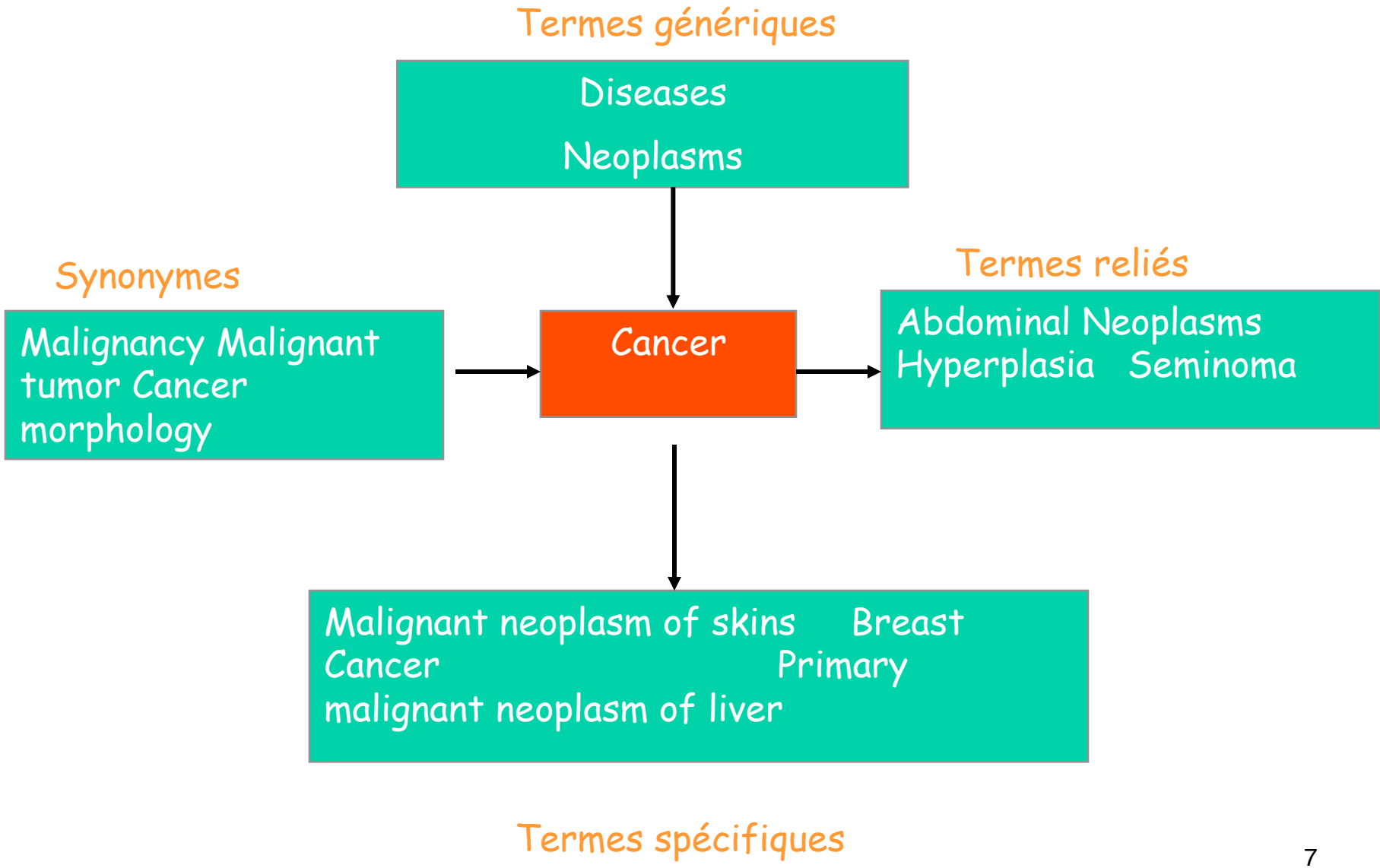
Société
Enfants, Gastronomie, Religion...

Sciences et technologies
Animaux, Astronomie, Physique...

Sciences humaines
Archéologie, Histoire, Économie...

Sites de : **Algérie** - **Belgique** - **Canada** - **Maroc** - **Suisse** - **Tunisie**
Nouveaux sites - **Sites de la semaine** - **Suggérer un site**

Exemple thésaurus

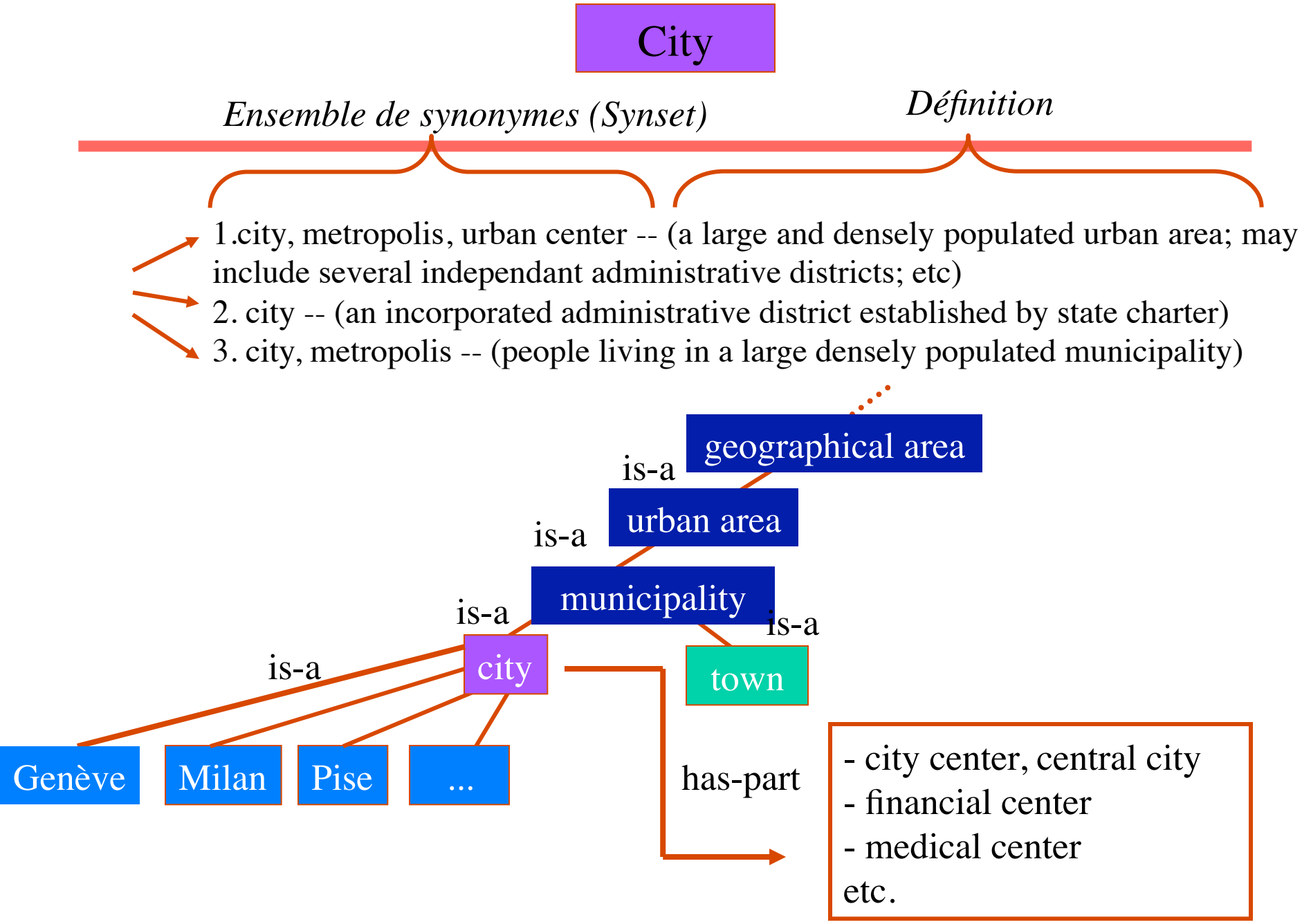


Exemple : WordNet

- Gratuit sur Internet

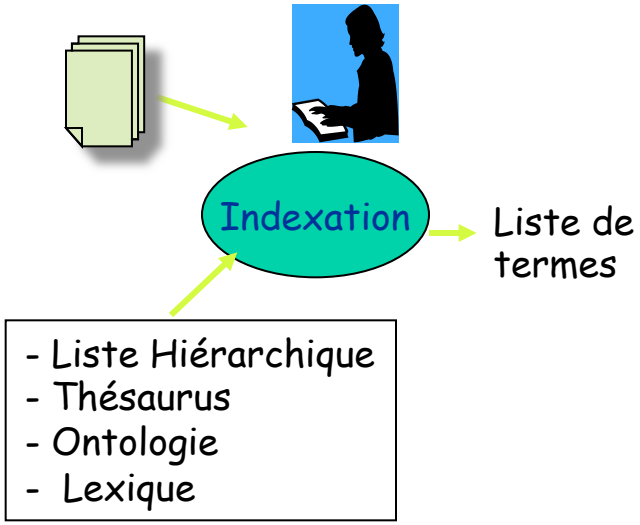
Exemple : WordNet

- Contenu : mots anglais
  - Organisés en synsets : ensembles de mots synonymes
    - Un mot peut appartenir à plusieurs synsets
    - Relations sémantiques entre synsets :
      - Hyperonymie/hyponymie(généralisation/spécialisation) (is-a),
      - antonymie (opposé à)
      - etc.


City
Ensemble de synonymes (Synset)
Définition
1.city, metropolis, urban center -- (a large and densely populated urban area; may include several independant administrative districts; etc)
2. city -- (an incorporated administrative district established by state charter)
3. city, metropolis -- (people living in a large densely populated municipality)
geographical area
is-a
urban area
is-a
municipality
is-a
city
is-a
town
is-a
Genève
Milan
Pise
...
has-part
- city center, central city
- financial center
- medical center
etc.

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 : Inconvénients 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
  - Statistique (distribution des mots) et/ou TALN (compréhension du texte)
  - Approche courante est plutôt statistique avec des hypothèses simples
    - Redondance d'un mot marque son importance
    - Cooccurrence des mots marque le sujet d'un document

Indexation automatique

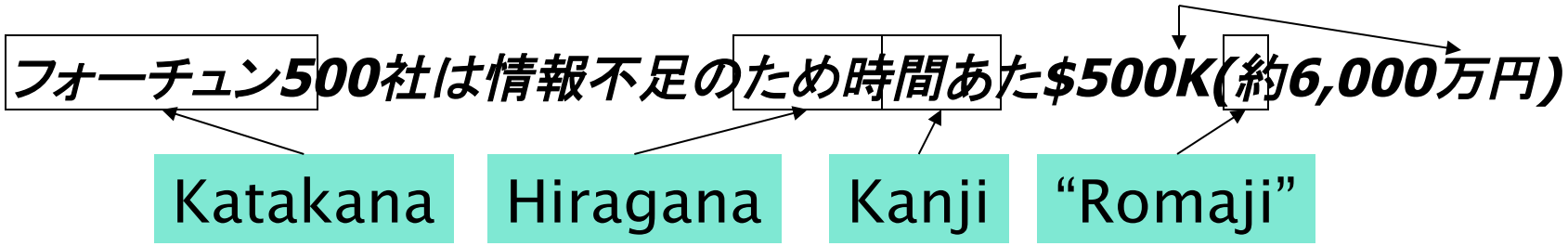
- Les mots-clés représentatifs du contenu des textes sont déterminés selon trois méthodes :
  - extraction de mots simples
  - analyse linguistique
  - analyse statistique
- Approche courante, plutôt statistique: 3 étapes
  - Étape 1 : extraction de mots simples
  - Étape 2 : normalisation des mots extraits
  - Étape 3: pondération des mots normalisés

Indexation automatique
Etape 1 : Extraction des mots (1)

- 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
  - Langue française
    - ***L'ensemble*** → un terme ou deux termes ?
      - ***L*** ? ***L'*** ? ***Le*** ?
  - Langue Allemande les mots composés ne sont pas segmentés
    - Lebensversicherungsgesellschaftsangestellter
    - 'life insurance company employee'

Etape 1 : Extraction des mots – (2)

- Pas d'espaces en chinois et en japonais
  - Ne garantit pas l'extraction d'un terme de manière unique
- Japonais encore plus compliqué avec différents alphabets



L'utilisateur peut exprimer sa requête entièrement en Hiragana

Etape 1 : Extraction des mots – (suite.)

- La langue arabe s'écrit de droite à gauche avec certains items écrits de gauche à droite (ex : les chiffres)
- Les mots sont séparés mais les lettres sont liées dans un mot

  - سبانيا "زلات" دبلوماسيته 2006
    الجزائر الحكومة الإسباني

- Orthographe des noms ex: Einstein peut s'écrire :

  اينشتاين; اينشطين , اينشتين

Etape 1 : Extraction des mots – (suite.)

- Suppression des mots « vides » (stoplist / Common Words removal)
  - Mots trop fréquents mais pas utiles
  - Exemples :
    - Anglais : the, or, a, you, I, us, …
    - Français : le , la de , des, je, tu, …
  - Attention à :
    - US : «USA » ; « give us information »
    - a de (vitamine a)

Etape : Normalisation

- «Lemmatisation» (radicalisation) / (stemming)
  - Processus morphologique permettant de regrouper les variantes d'un mot
    - Ex : économie, économiquement, économiste, ⇨ économ
    - pour l'anglais : retrieve, retrieving, retrieval, retrieved, retrieves ⇨ retriev

Forme morphologique d'un mot

Etape 2 : Normalisation (suite.)

- Utilisation de règles de transformations
  - règle de type : condition action
    - Ex : si mot se termine par s supprimer la terminaison
  - Technique utilisée principalement pour l'anglais
    - L'algorithme le plus connu est : Porter
- Analyse grammaticale
  - Utilisation de lexique (dictionnaire)
  - Tree-tagger (gratuit sur le net)
- Troncature

Normalisation : Algorithme de Porter

- Basée sur la mesure de séquences voyelles-consonnes
  - mesure m pour un «stem» est [C](VC)^m[V] ou C est une séquence de consonnes et V est une séquence de voyelles [] = option
  - m=0 (tree, by), m=1 (trouble,oats, trees, ivy), m=2 (troubles, private)

- Algorithme basé sur un ensemble de conditions actions
  - old suffix → new suffix
  - Les règles sont divisées en étapes et sont examinées en séquence
  - e.g. Step 1a:
    - sses → ss *(caresses → caress)*
    - ies → i *(ponies → poni)*
    - s → NULL *(cats → cat)*
  - e.g. Step 1b:
    - if m>0 eed → ee *(agreed → agree)*
    - if *v*ed → NULL *(plastered → plaster but bled → bled)*

- Plusieurs implantations sont accessibles
  - http://www.tartarus.org/~martin/PorterStemmer/

Porter algorithm

(Porter, M.F., 1980, An algorithm for suffix stripping, *Program*, **14**(3) :130-137)

- Step 1: plurals and past participles
  - SSES -> SS caresses -> caress
  - (*v*) ING -> motoring -> motor
- Step 2: adj->n, n->v, n->adj, …
  - (m>0) OUSNESS -> OUS callousness -> callous
  - (m>0) ATIONAL -> ATE relational -> relate
- Step 3:
  - (m>0) ICATE -> IC triplicate -> triplic
- Step 4:
  - (m>1) AL -> revival -> reviv
  - (m>1) ANCE -> allowance -> allow
- Step 5:
  - (m>1) E -> probate -> probat
  - (m > 1 and *d and *L) -> single letter controll -> control

Normalisation : Algorithme de Porter Exemple

- Texte original:

  marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales

- Texte après porter + suppression mots vides:

  Market 4, strateg 1, carr 1, compan 1, US 1, agricultur 1, chemic 2, report 2, predict 2, share 1, statist 1, agrochem 1, pesticid 1, herbicid 1, fungicid 1, insecticid 1, fertil 1, sale 2, stimul 1, demand 1, price 1, cut 1, volum 1

Etape : Normalisation par troncature

- Tronquer les mots à X caractères
  - Tronquer plutôt les suffixes
  - Exemple troncature à 7 caractères
    - économiquement : écomoni

Quelle est la valeur optimale de X ? : 7 caractères pour le Français

Indexation automatique
Exemple

- Texte : un système de recherche d ’informations (document) (SRI, base de données documentaires) permet d ’analyser, d ’indexer et de retrouver les documents pertinents répondant à un besoin d ’un utilisateur.

- système, recherche, informations, document, SRI, base, données, documentaires, analyser, indexer, retrouver, documents, pertinents, répondant, besoin , utilisateur

- systeme, recherc, informa, documen, sri, base, donnee, documen, analyse, indexer, retrouv, documen, pertine, reponda, besoin, utilisa

- systeme 1,recherc 1, informa 1, documen 3, sri 1, base 1, donnee 1, analyse1, indexer 1, retrouv 1, pertine 1, reponda 2, besoin 3, utilisa *1*

Inconvénients de la normalisation (1)

- Les algorithme de “Stemers” sont souvent difficiles à comprendre et à modifier
- Peut conduire à une normalisation “agressive”
  - Exemple :
    - policy/police, execute/executive, university/universe, organization/ organ sont normalisés Porter
    - Internet/Interne (troncature)
- Oublis de quelques normalisations intéressantes
  - Exemple :
    - European/Europe, matrices/matrix, machine/machinery ne sont pas normalisés

Inconvénients de la normalisation (2)

- Produit des “stems” qui n’ont pas de sens donc difficiles à interpréter
  - Exemple
    - avec Porter, “iteration” produit “iter” et “general” produit “gener”

- Il existe des techniques (analyse de corpus) pour réduire ces effets négatifs.

Est-il nécessaire de reconnaître un mot, une racine ?

- La méthode des n-grammes
  - Définition : un n-gram est une succession de n lettres.
  - Généralement n = 1,2,3
  - Exemple : retrieval
    - 1-gram : r, e, t, r, i, e, v, a, l
    - 2-gram : re, et, tr, ri, ie, ev, va, al
    - 3- gram : ret, etr, tri, rie, iev, eva, val
  - Utilisée pour le chinois
  - Intéressant pour la radicalisation

Exemple

- Comparer retrieve et retrieval
  - A=retrieve :
    - ret, etr, tri, rie, iev, eve
  - B=retrieval
    - ret, etr, tri, rie, iev, eva, val

- Sim (A,B)=2*nb_comm/(nb_A +nb_B)

Fichier inverse

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

Organisation du fichier inverse

Dictionnaire

Mot	Nb Doc	FrqTotal	Ptr
Ambitious	2	6	1
Brutus	2	4	3
capitol	5	15	6

Posting simple

doc	Freq
doc1	3
doc2	2
doc1	1
doc3	7

- Liste triée
- B-Arbre
- Table de hashage (hash-code)
- ...

Position du terme dans le document (important pour la recherche d'expressions)

Posting riche

doc	Freq	position	balise
doc1	3	1, 4, 3	1, 5
doc2	2	1	
doc3	2	3	
	0	0	

Balises (title, body, anchor, ..)

Organiser les termes et les documents dans un Fichier Inverse

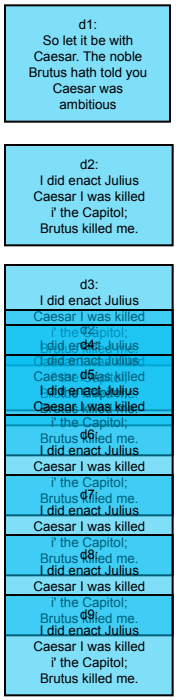
d1:
So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

d2:
I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Traitement = Indexation

Term	N docs	Tot Freq	Ptr
ambitious	1	1	1
be	1	1	2
brutus	2	2	3
capitol	1	1	5
caesar	2	3	6
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



Répondre à une demande (requête)

caesar

Term	N docs	Tot Freq	Ptr
ambitious	1	1	1
be	1	1	2
brutus	2	2	3
capitol	1	1	5
caesar	2	3	6
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

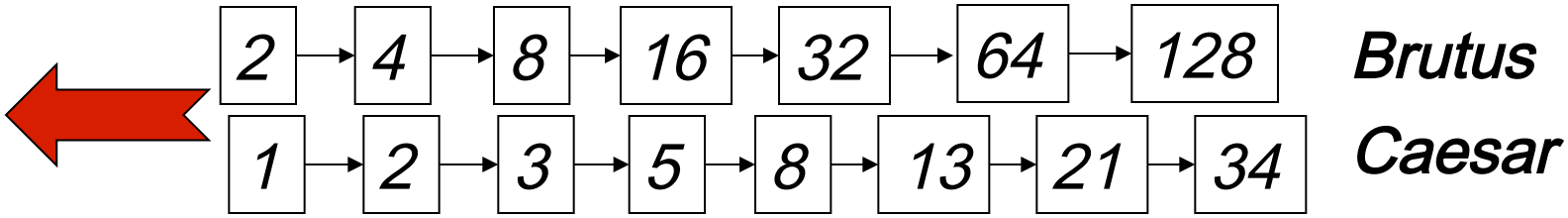
d1:
So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

d2:
I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

d3:
I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Trait. requête

- Soit la requête :
  - ***Brutus*** *AND* ***Caesar***
  - Chercher ***Brutus*** dans le dictionnaire;
    - Sélectionner sa liste postings.
  - Chercher *Caesar* dans le dictionnaire ;
    - Sélectionner sa liste postings.
  - "Fusion" des deux postings:

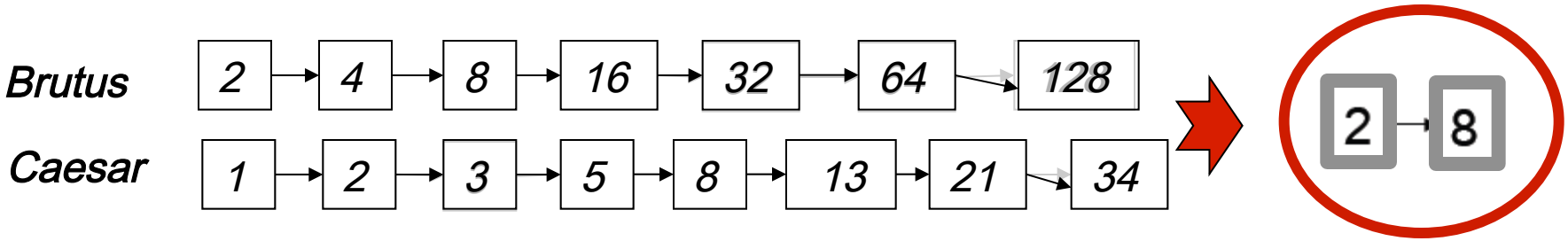


Fusion

- Parcourir les deux *postings* simultanément

Les deux listes sont **ordonnées**
→
Si les longueurs des listes sont x et y, l'algo est en O(x +y)

```
fusion = <>
id1 = l1[0], id2 = l2[0]
Tant que les listes ne sont pas vides
    si id1 = id2 alors
        ajouter(fusion, id1)
        id1 = suivant(l1)
        id2 = suivant(e2)
    sinon
        si id1 < id2 alors
            id1 = suivant(l1)
        sinon
            id2 = suivant(l2)
```



Démarche de construction d'un fichier inverse

- La construction d'un fichier inverse est une «étape importante
- Elle peut prendre énormément de temps

Extraire les mots de chaque document

- 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

Trier le fichier termes-documents (1)

- 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

Trier le fichier termes-documents (2)

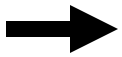
- Pour chaque terme,
  - on dispose de la liste de documents qui le contient
  - Le nombre de documents comportant ce terme

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

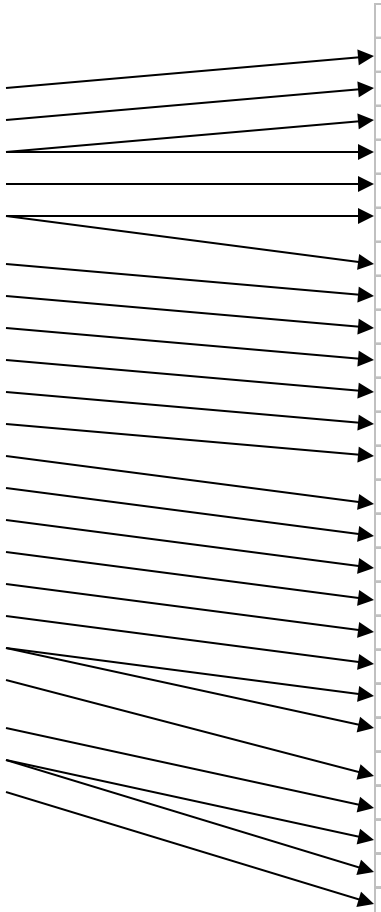
Term	Doc #	Freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

Construire le dictionnaire et le « posting »

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

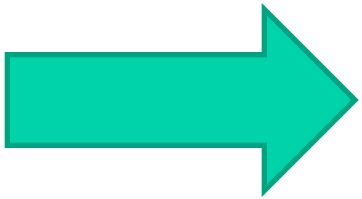


Doc #	Freq
2	1
2	1
1	1
2	1
1	1
1	1
2	2
1	1
1	1
2	1
1	2
1	1
2	1
1	1
1	2
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1

Traitement de collections volumineuses

Terme	*Id. 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

rme	*Id. Doc*
	2
	2
	2
	2
th	2
esar	2
e	2
ble	2
utus	2
th	2
d	2
u	2
caesar	2
was	2
ambitious	2



Tri par termes

(puis par documents)

Terme	*Id. 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
.....	.....

Tri

- Généralement effectué en **mémoire vive**
- Mais, … impossible pour les collections volumineuses
  - On analyse un document à la fois (compression très difficile)
  - Les listes des index ne sont complètes qu'à la fin du processus
- Solution → tri par bloc « diviser pour mieux régner »
  - Diviser la collection en *n* parties gérables en mémoire
  - Trier chaque partie séparément, et réécrire le résultat sur le disque
  - Fusionner les résultats (les fichiers) 2 par 2.

Indexation distribuée

- Pour de très larges collections (Web)
- Un serveur principal dirige le tout (doit être très sûr)
- 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

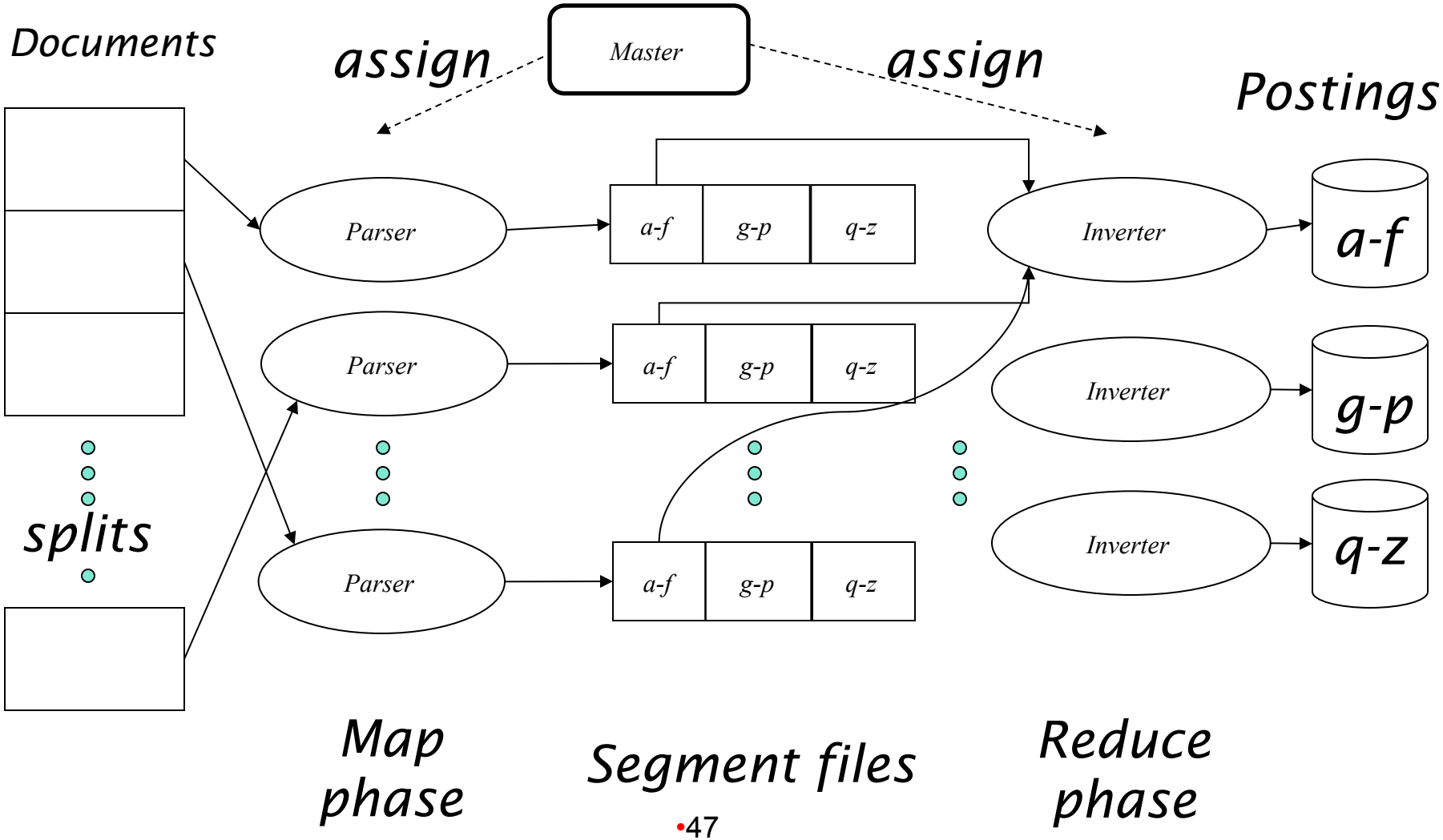
  (Google File System & Map Reduce)
- Implémentation libre développée dans le projet Hadoop 

MapReduce

- Principe :
  - Tous les algos 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*

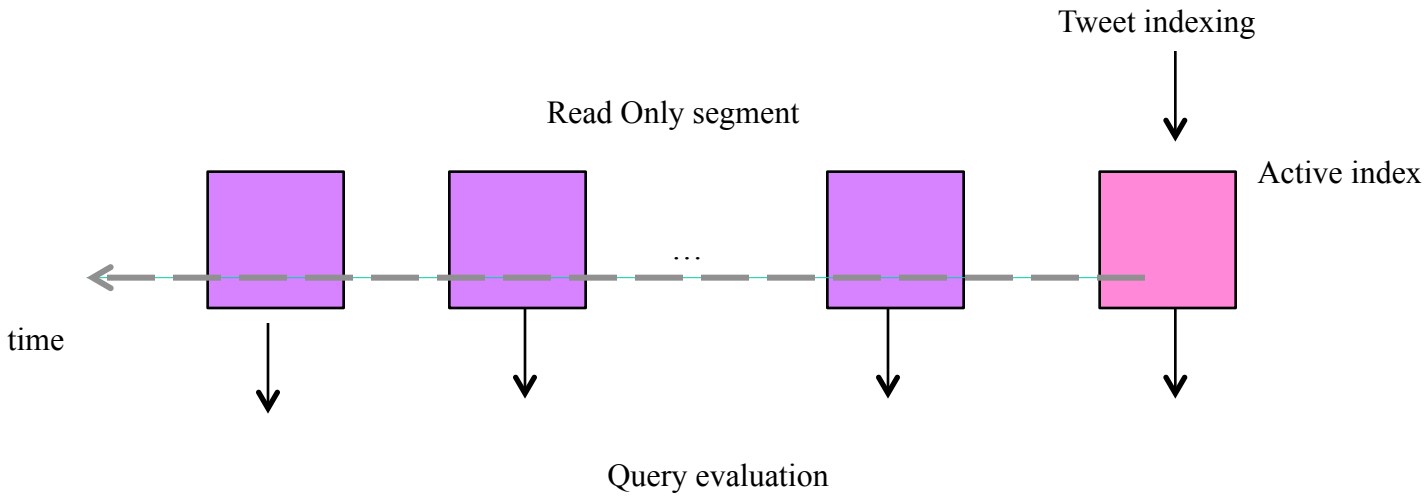
- Interêt :
  - Les tâches *map* sont exécutées sur les machines sur lesquelles sont stockées les données
  - Les tâches *map* sont exécutées en parallèle

Construction de l'index avec MapReduce

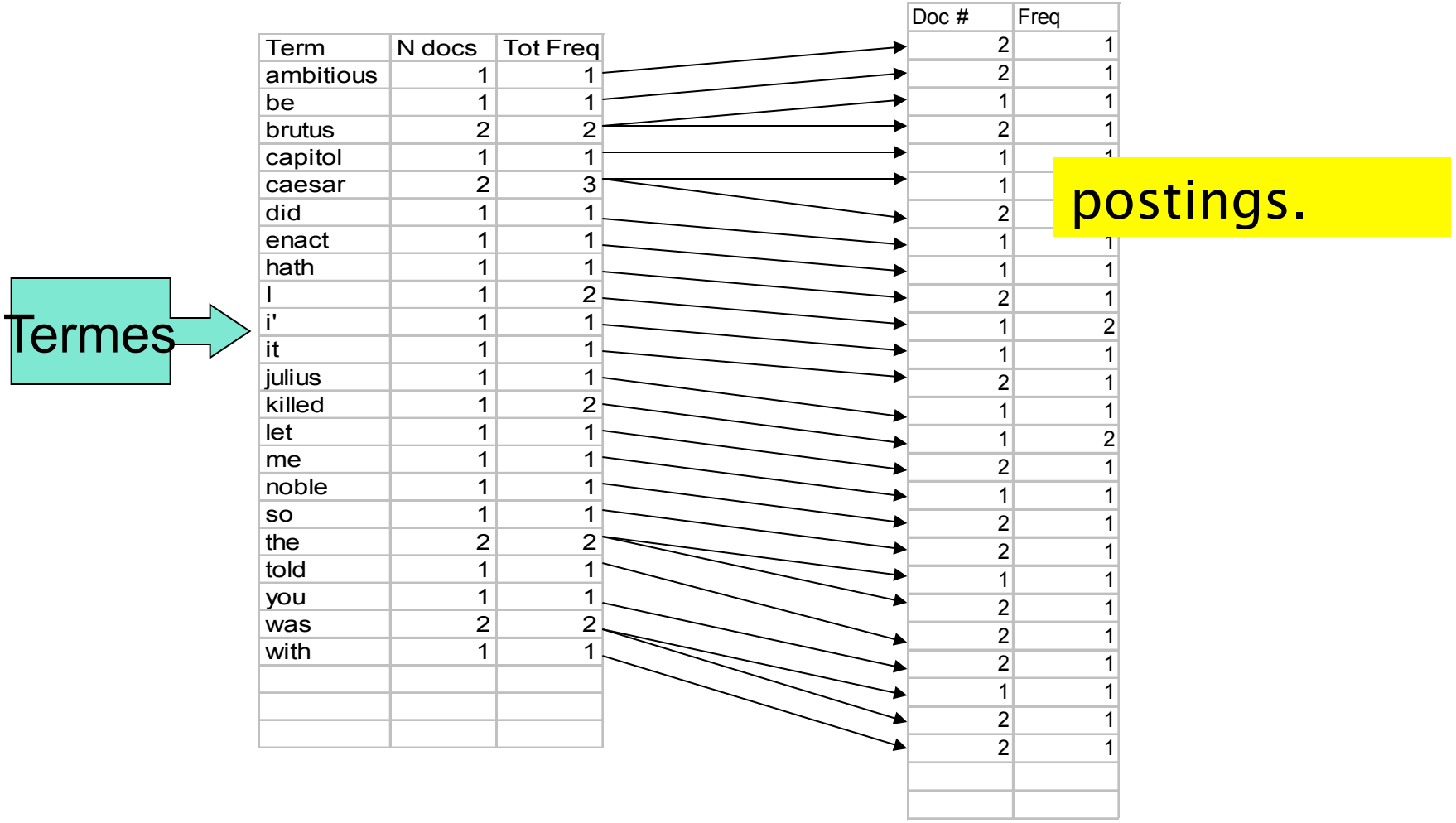


Indexation en temps réel

- Real-time indexing
  - Selective indexing (Chen et al SIGMOD'11
    - Limit the number of tweets to be indexed (only 20% of queries represent 80% of user requests
  - Earlybird (Bush et al ICDE'12)
    - Low query latency (50ms)
    - Tweets are searchable within 10 seconds



Coût du stockage



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

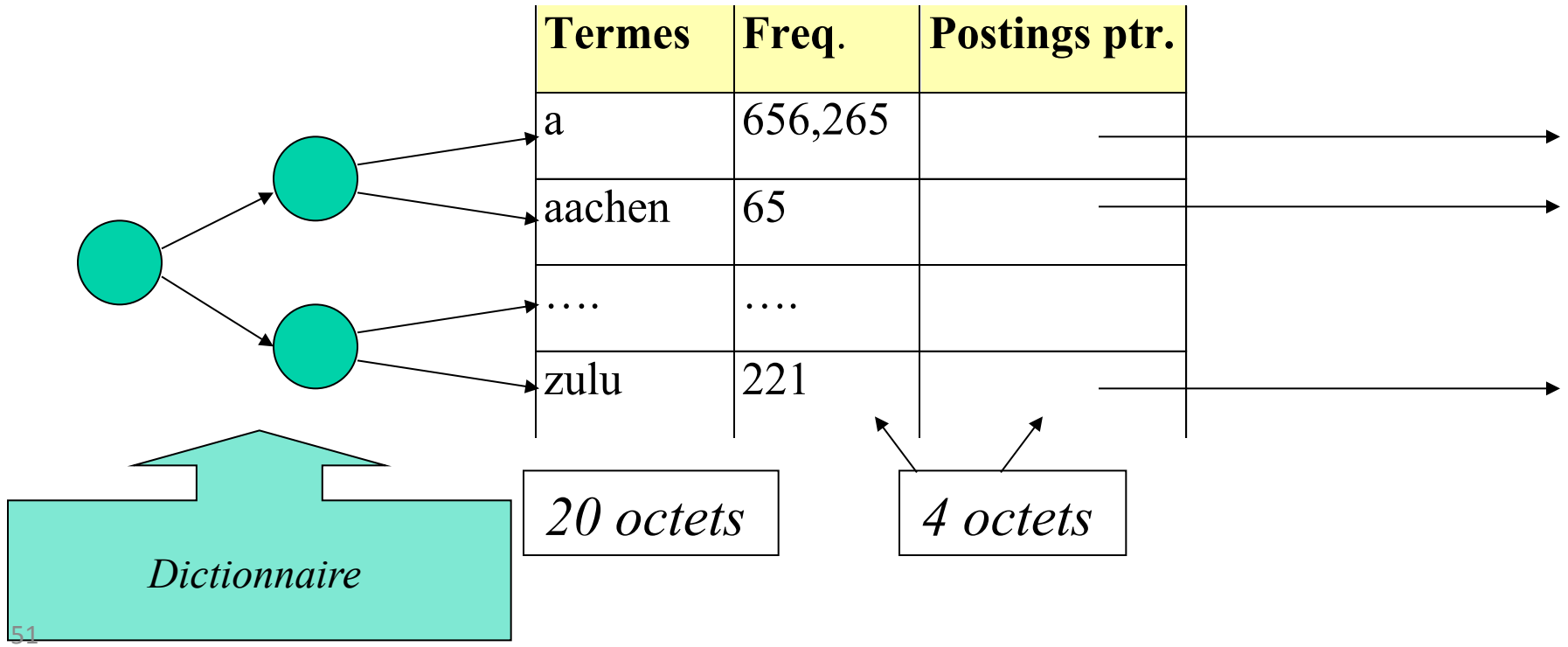
Doc #	Freq
2	1
2	1
1	1
2	1
1	1
1	1
2	1
1	1
1	1
2	1
1	2
1	1
2	1
1	1
1	2
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1

Réduction du coût de stockage → Compression

Compression du dictionnaire

Taille du dictionnaire

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



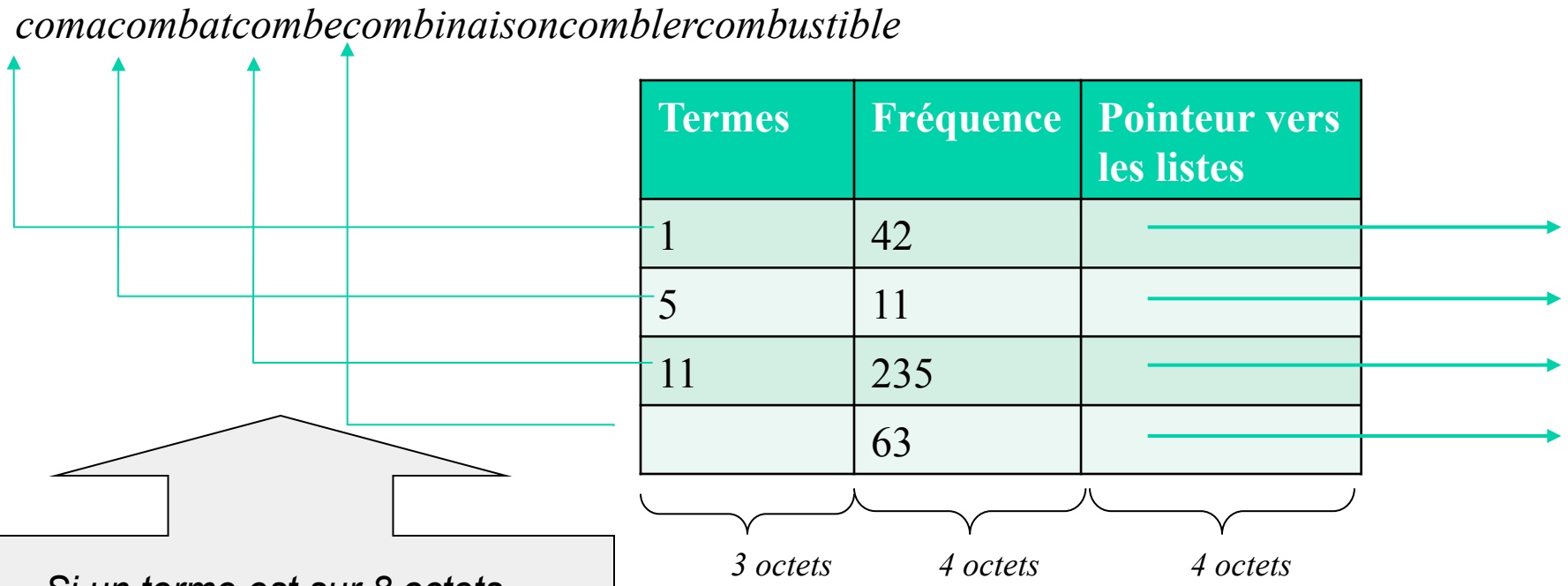
Beaucoup d'espace perdu

- 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 » « *supercalifragilisticexpialidocious* » ou « *hydrochlorofluorocarbons* ».

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



Si un terme est sur 8 octets,
le dictionnaire est sur 11 octets
→ 19 octets par terme au lieu de 28
400K termes x 19 ⇒ 7.6 MB

Exercice : determiner la taille optimale du pointer pour 400 Mille termes

Compression de la liste de termes : pointeurs par Blocs

- Stocker les pointeurs à chaque k termes (Exemple : k=4).
- Besoin de rajouter un octet pour stocker la taille du terme

4coma6combat5combe11combinaison7combler11combustible

Termes	Fréquence	Pointeur vers les listes
	42	
	11	
	235	
	63	

3 octets *4 octets* *4 octets*

- On économise 3 pointeurs (9 octets) tous les k=4 termes.
- On dépense 1 octet de plus à chaque mot pour la taille
- On utilise 3+4 octets au lieu 3X4 octets, → économie de 0.5MO
- on réduit la taille → 7,1 Mo

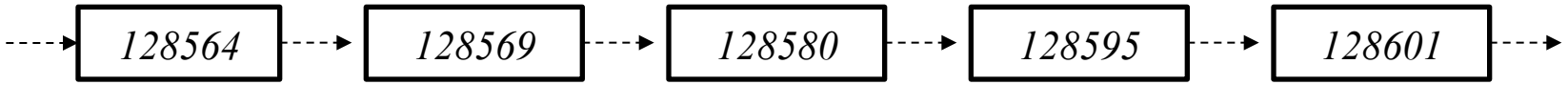
Exercice

- Pourquoi ne pas augmenter k?
- Estimer l'espace nécessaire pour l'index (ce que l'on gagne vis-à-vis des 7,6 MO) pour *k = 4, 8* et *16.*

Compression du *posting*

Compression du *posting* (listes de documents)

- Le fichier *Posting* est au moins 10 fois plus volumineux que le dictionnaire
  - Le *Posting* est formé de DocId(s) (numéro de document) → un entier codé sur 4 octets
  - Au mieux, pour 1 M de documents, sur $\log_2$ 1 000 000 ≈ 20 bits
- Peu de termes fréquents, beaucoup de termes rares
  - « ***arachnocentric*** » apparaît peut-être une fois dans toute la collection → 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* (listes de documents)

- Les docid(s) du *posting* sont stockés par ordre croissant
  - ***computer***: 33,47,154,159,202 …
- Stocker l'écart (intervalle) entre les docid(s) au lieu des docids .
  - 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)

Compression du posting (listes de documents)

Ex. trois entrées de postings

	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								

20 bits, ça reste excessif pour "the"

Compression du posting

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

- → Pour une valeur d'écart l, on veut utiliser aussi peu de bits possible (l'entier au-dessus de $\log_2 l$).
- En pratique, on arrondit à l'octet supérieur

- → 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 $l \leq 127$, 7 bits suffisent → $c = 1$.
  - Sinon, $c = 0$ et on continue sur l'octet suivant.
  - $c = 1$ signifie toujours que le nombre se termine à cet octet.

Exemple

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

Postings stockés comme concaténattion d'octets

00000110101110001000010100001101000011001011000 1

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

Qualité de l'indexation

- Exhaustive*(cf. rappel)*
  - Complétude, nombre d'éléments (sujets, concepts) indexés
  - Limiter le silence
- Spécificité *(cf. précision)*
  - Exactitude (Précision) des index
  - Limiter le bruit

Conclusion : Représentation de l'information

- 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 + statistique

Récapitulatif : Représentation de l'information

- 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.

Exercices

- Ecrire un algorithme d'indexation permettant la construction du fichier inverse ?

- Structure du fichier inverse pour l'utilisation des expressions