Recherche d'information textuelle
Modèles

Anne-Laure Ligozat

2018/2019 [1]

1. librement inspiré des cours de Benjamin Piwowarski et de Hinrich Schütze

Systèmes de RI

- Indexation des documents
- Analyse de la requête
- Modèle de recherche
- Évaluation

Plan

Pourquoi indexer ?

- Objectif : trouver documents pertinents pour la requête utilisateur
- À partir des **mots** de la requête
- Parcours complet de la collection impossible
  - trop de documents → temps de réponse prohibitif
  - opérations entre termes (not, near...) complexes

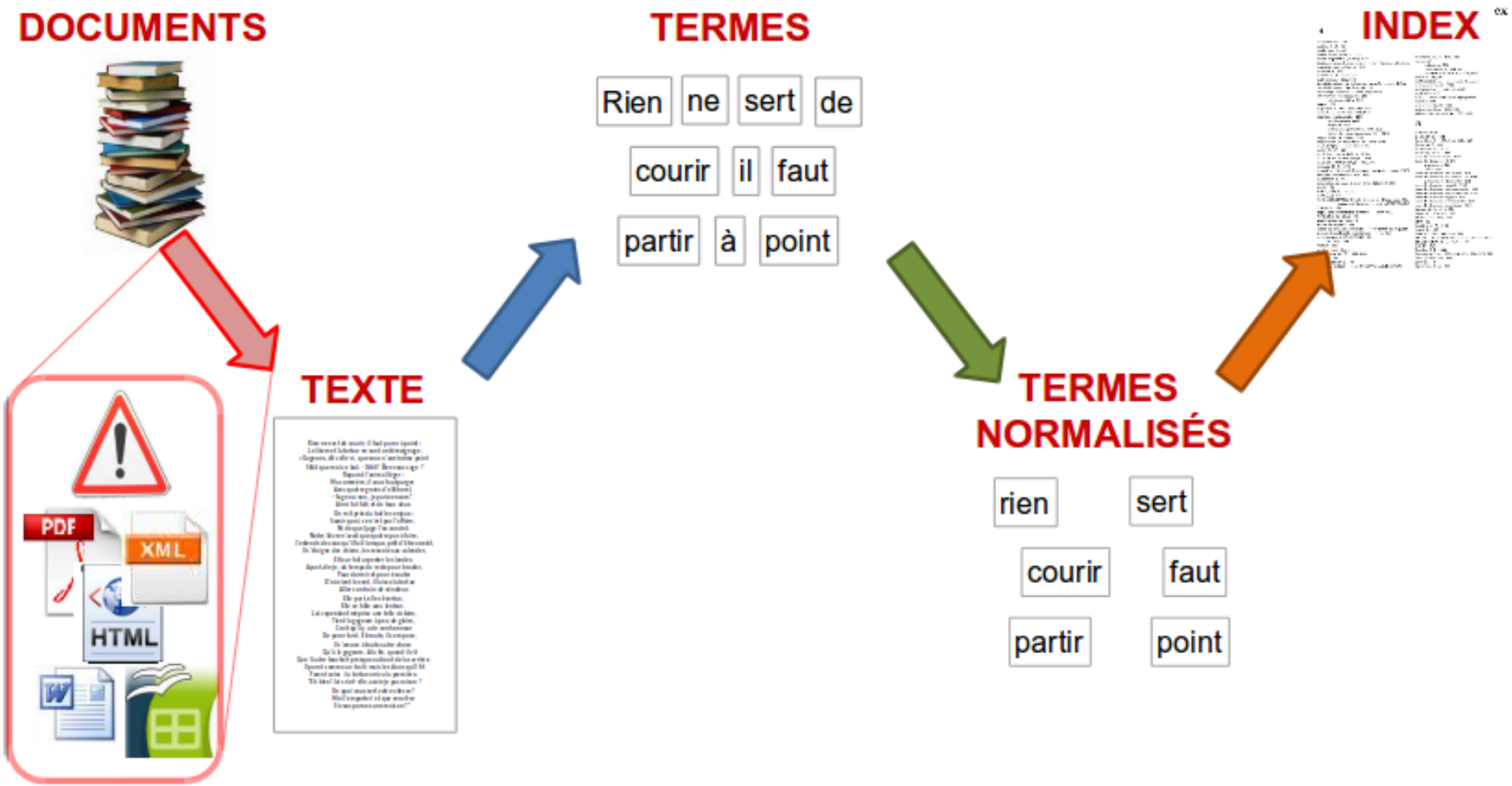
⇒ traitement préalable = indexation

- but : "transformer des documents en substituts capables de représenter le contenu de ces documents" (Salton et McGill, 1983)

Indexation libre vs contrôlée

- Indexation libre : termes des documents
- Indexation contrôlée : termes prédéfinis
  - vocabulaire contrôlé : évite polysémie, synonymie, problèmes de granularité

Documents


DOCUMENTS
TERMES
INDEX
Rien ne sert de
courir il faut
partir à point
TEXTE
TERMES NORMALISÉS
rien sert
courir faut
partir point
PDF
XML
HTML

Formats

Formats

- HTML (diff : menus, tableaux, publicité, rendu)
- texte brut (structure ?)
- pdf (encodage, rendu)
- word (format propriétaire, structure)
- excel (gestion des tableaux)
- openoffice (xml)

Prise en compte du format

- détecter le type d'un document est assez simple
- heuristiques spécifiques à chaque format pour extraire le texte
- les moteurs de recherche utilisent très rarement la structure des documents

Langue et encodage

Langues

- identification de langue(s) = problème difficile
- recherche d'information multilingue possible

Encodages

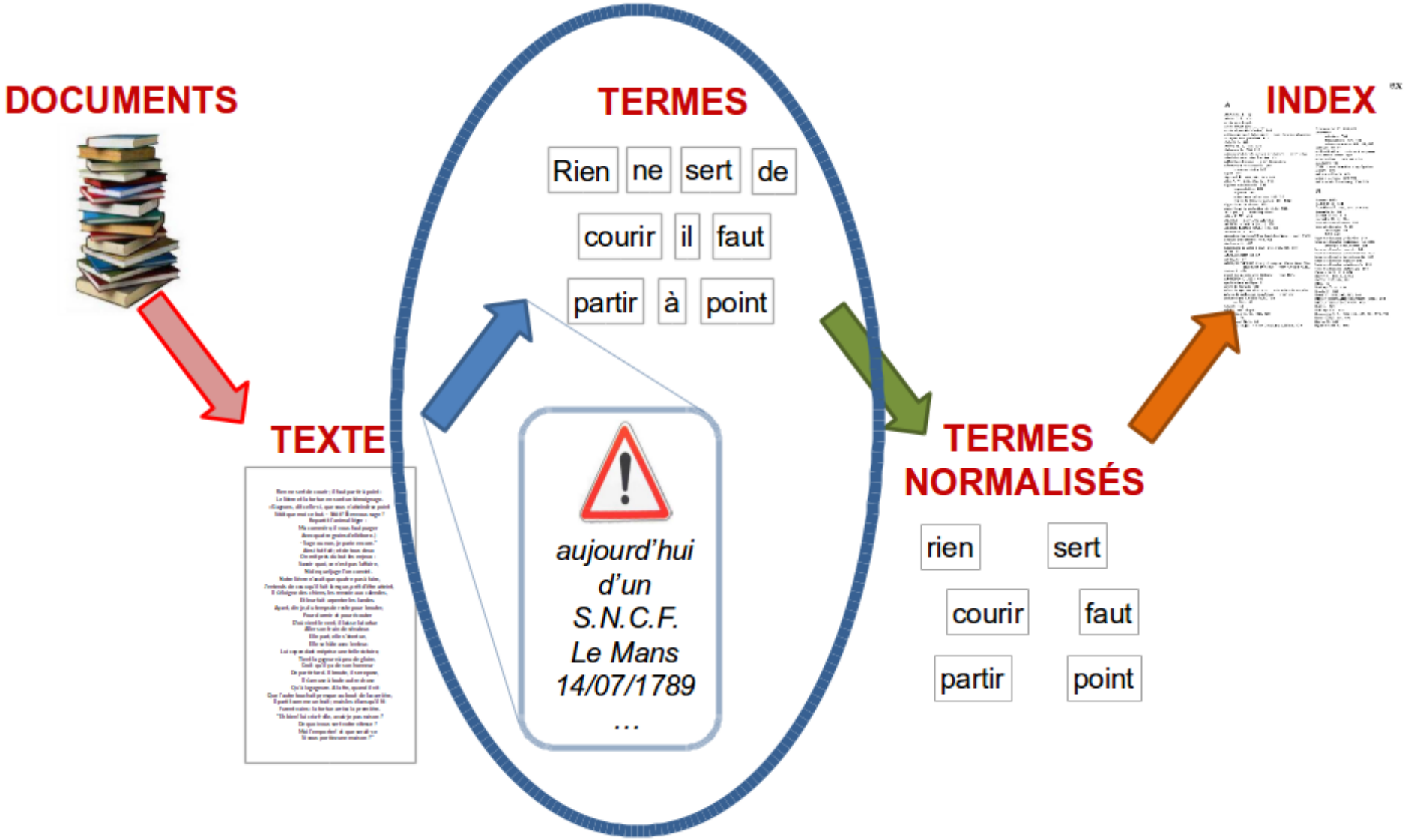
- erreurs dans gestion d'encodage → résultats erronés

Contenu des documents

unité =

- fichier ?
- e-mail ?
  - avec entêtes ?
  - avec attachements ?
- ensemble de fichiers
  - site web
  - documents en plusieurs fichiers
- ...

Du texte aux termes



Segmentation

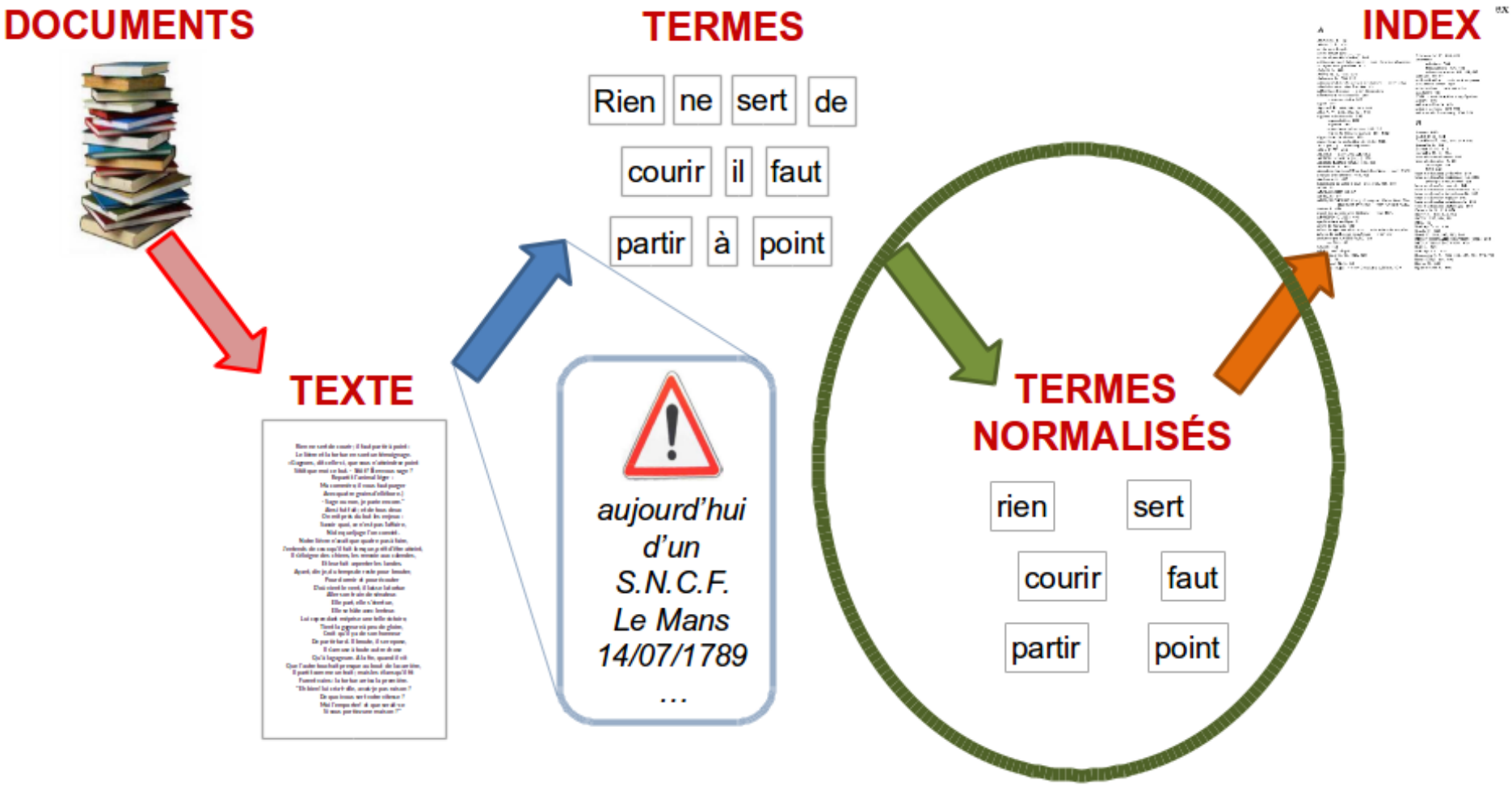
Segmentation = identification des unités élémentaires

- **mot** : chaîne de caractères telle qu'elle apparaît dans le texte
- **terme** (ou type) : mot normalisé (casse, morphologie, orthographe...)
  - ensemble des termes = dictionnaire
- **token** : instance d'un mot ou terme dans un document

Difficultés de la segmentation en tokens

- variantes graphiques des mots avec séparateurs possibles
  - États-Unis ou États Unis
- mots composés des langues agglutinantes
  - Lebensversicherungsgesellschaftsangestellter (employé d'une compagnie d'assurance-vie)
- alphabets multiples en japonais par exemple
- bidirectionnalité du sens de lecture en arabe (chiffres et lettres)
- nombres : 555 3424, 24.09.2018
- ...

Normalisation



Normalisation de variantes (1/2)

- dans les documents et dans la requête
- variantes à regrouper
  - variantes de mots incluant ponctuation
    - U.S.A. et USA
    - morpho-syntaxe et morphosyntaxe
  - variantes diacritiques
    - en allemand, Tuebingen, Tübingen et Tubingen
    - en anglais, resume = résumé
  - variantes de noms propres
    - Gorbatchov et Gorbatchev
- mais
  - accents peuvent être pertinents
    - sur et sûr
    - pêche et péché
  - casse peut être pertinente
    - en allemand, mit et MIT
      (interaction entre normalisation et détection de langue)
    - en anglais, fed et Fed

Normalisation de variantes (2/2)

- possiblement asymétrique
  - window → window, windows
  - windows → Windows, windows
  - Windows : pas d'expansion
- + fautes de frappe ou d'orthographe, erreurs OCR
- critère important : comment les utilisateurs écriront-ils leur requête le plus souvent ?

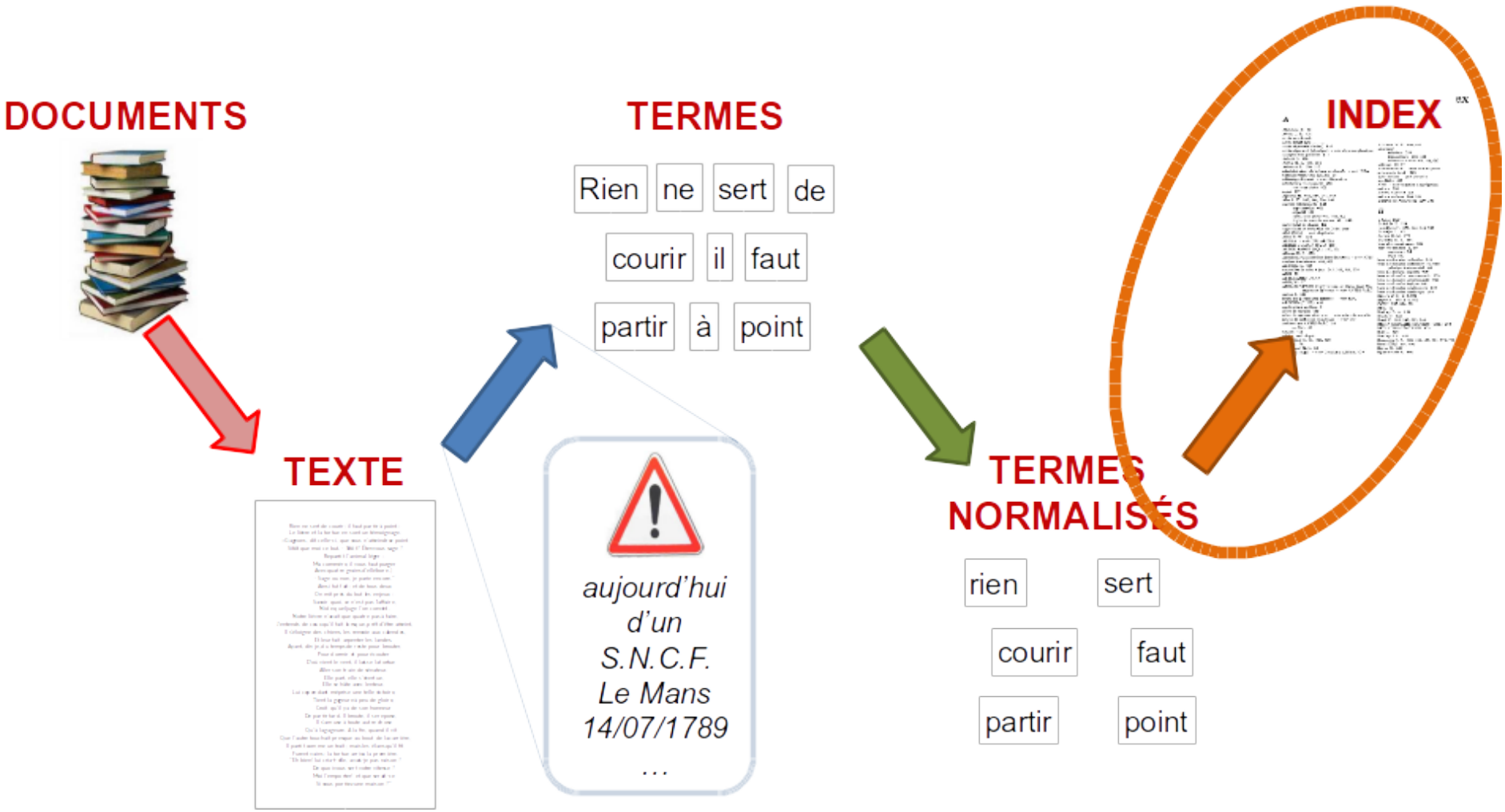
Normalisation morphologique

- utilisation d'analyses
  - lemmatisation : chanteurs → chanteur, chantions → chanter
  - racinisation : automate, automatique, automatisation → automat
    - notamment algorithme de Porter, algorithme classique pour l'anglais
    - racinisation utile pour certaines requêtes, dégrade résultats pour d'autres
  - étiquetage
- techniques assez bien maîtrisées : pourcentage d'erreurs faible mais difficilement compressible

Mots "vides" (stop words)

- mots "outils" n'apportent pas de sens au texte
  - déterminants : le, la
  - pronoms : je, nous
  - prépositions : sur, contre
- ce sont les plus fréquents de la langue
  - les 30 mots les plus fréquents représentent 30% des occurrences de mots
  - les supprimer permet d'économiser beaucoup de place dans l'index
- mais
  - utiles pour requêtes multi-termes : "pomme de terre", "les Chevaliers du Zodiaque"
  - parfois porteurs de sens dans des cas particuliers : "Let it be", "The Who", "être ou ne pas être"
  - compression permet finalement de conserver les mots vides dans peu d'espace

Index


DOCUMENTS
TERMES
INDEX
Rien ne sert de
courir il faut
partir à point
TEXTE
aujourd'hui
d'un
S.N.C.F.
Le Mans
14/07/1789
...
TERMES
NORMALISÉS
rien sert
courir faut
partir point

Matrice d'incidence

	Antoine & Cléopâtre	Jules César	La Tempête	Hamlet	Othello	Macbeth
Antoine	1	1	0	0	0	0
Brutus	1	1	0	1	0	0
César	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cléopâtre	1	0	0	0	0	0
pitié	1	0	1	1	1	1
pire	1	0	1	1	1	0

Matrice d'incidence

Brutus ET Cléopâtre ET PAS Calpurnia

	Antoine & Cléopâtre	Jules César	La Tempête	Hamlet	Othello	Macbeth
Antoine	1	1	0	0	0	0
Brutus	1	1	0	1	0	0
César	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cléopâtre	1	0	0	0	0	0
pitié	1	0	1	1	1	1
pire	1	0	1	1	1	0

Vecteurs d'incidence

¬Calpurnia	1	0	1	1	1	1

ET "bit à bit"

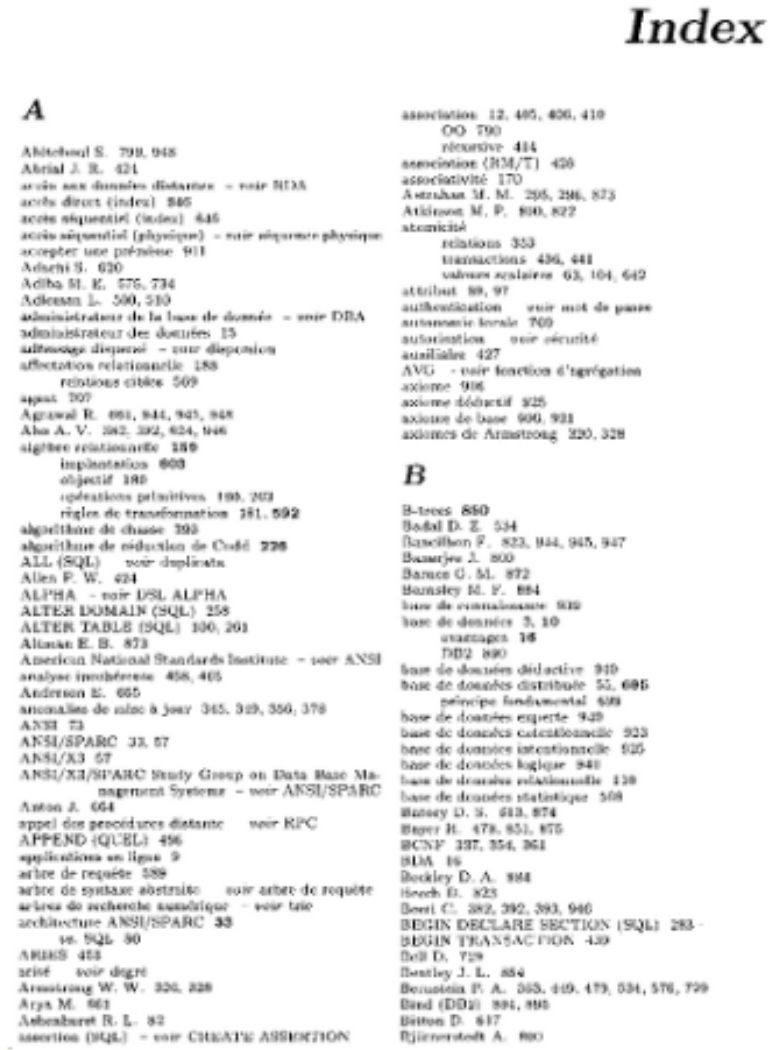
1	0	0	0	0	0

Matrice d'incidence

- impossible à utiliser en pratique
  - collection d'un million de documents
  - environ 1000 mots par document en moyenne
  - vocabulaire total de 500 000 mots distincts

  → combien de cases dans la matrice ? combien de 1 ? combien de 0 ?

Fichier inverse



Index

A

Abiteboul S. 798, 948
Abrial J. R. 424
accès aux données distantes – voir NDA
accès direct (index) 846
accès séquentiel (index) 846
accès séquentiel (physique) – voir séquence physique
accepter une prémisse 901
Adiba S. 620
Adiba M. E. 575, 734
Adleman L. 500, 510
administrateur de la base de donnée – voir DBA
administrateur des données 15
affinage disjoint – voir disjonction
affectation relationnelle 188
  relations citées 509
agent 707
Agrawal R. 661, 844, 945, 948
Aho A. V. 382, 392, 624, 946
algèbre relationnelle 189
  implantation 603
  objectif 189
  opérations primitives 190, 203
  règles de transformation 191, 592
algorithme de chasse 393
algorithme de réduction de Codd 226
ALL (SQL) voir duplicata
Allen F. W. 424
ALPHA – voir DSL ALPHA
ALTER DOMAIN (SQL) 258
ALTER TABLE (SQL) 100, 261
Altman E. B. 873
American National Standards Institute – voir ANSI
analyse ascendante 458, 465
Anderson E. 665
anomalies de mise à jour 345, 349, 350, 378
ANSI 73
ANSI/SPARC 33, 57
ANSI/X3 57
ANSI/X3/SPARC Study Group on Data Base Management Systems – voir ANSI/SPARC
Anton J. 664
appel des procédures distante voir RPC
APPEND (QUEL) 456
applications en ligne 9
arbre de requête 588
arbre de syntaxe abstraite voir arbre de requête
arbres de recherche numérique – voir trie
architecture ANSI/SPARC 33
  vs. SQL 80
ARIES 458
arité voir degré
Armstrong W. W. 326, 328
Arya M. 661
Ashenhurst R. L. 82
assertion (SQL) – voir CREATE ASSERTION
association 12, 405, 406, 410
  OO 790
  récursive 414
association (RM/T) 420
associativité 170
Astrahan M. M. 295, 296, 873
Atkinson M. P. 800, 822
atomicité
  relations 353
  transactions 436, 441
  valeurs scalaires 63, 104, 642
attribut 89, 97
authentification voir mot de passe
autonomie locale 769
autorisation voir sécurité
auxiliaire 427
AVG – voir fonction d'agrégation
axiome 906
axiome déductif 325
axiome de base 906, 931
axiomes de Armstrong 320, 328

B

B-trees 850
Badal D. Z. 534
Bancilhon F. 823, 944, 945, 947
Banerjee J. 800
Banon G. 51, 872
Barnsley M. F. 884
base de connaissance 930
base de données 3, 10
  avantages 16
  DB2 880
base de données déductive 910
base de données distribuée 55, 695
  principe fondamental 699
base de données experte 929
base de données extensionnelle 923
base de données intentionnelle 925
base de données logique 940
base de données relationnelle 139
base de données statistique 508
Batory D. S. 613, 874
Bayer R. 478, 851, 875
BCNF 337, 354, 361
BDA 16
Beckley D. A. 884
Beech D. 825
Beeri C. 382, 392, 393, 946
BEGIN DECLARE SECTION (SQL) 283
BEGIN TRANSACTION 439
Bell D. 729
Bentley J. L. 884
Bernstein P. A. 365, 449, 479, 534, 576, 729
Bird (DDI) 894, 895
Bitton D. 617
Bjorneroteft A. 880

Fichier inverse

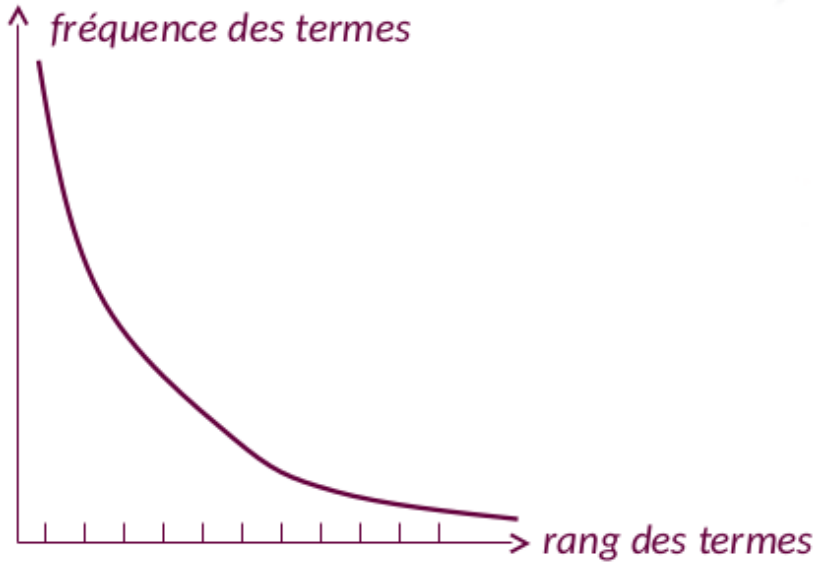
- notion classique de l'index
- associe des index aux documents qui les contiennent (identifiant unique)
  - a $\rightarrow$ d1, d2, d3, d4, d5...
  - à $\rightarrow$ d1, d2, d3, d4, d5...
  - abaissa $\rightarrow$ d3, d4...
  - abaissable $\rightarrow$ d5
  - abandon $\rightarrow$ d1, d5
  - abandonna $\rightarrow$ d2
  - ...

Taille du vocabulaire

- Le vocabulaire grandit quand la collection grandit
- Loi de Heaps : $M = kT^b$ avec
  - M taille du vocabulaire
  - T nombre de tokens dans la collection
  - b et k constantes (typiquement b=0,5 et k = 30 à 100)
  - loi empirique
- et c'est bien pire pour le web !

Fréquence des termes

- peu de mots fréquents et beaucoup de mots rares
- Loi de Zipf : le n^e mot le plus fréquent a une fréquence (= nb d'occurrences) proportionnelle à $1/n$



tfidf

- dans une requête comme dans un document, les termes n'ont pas tous la même importance
- intuition 1 : plus un document contient d'occurrences d'un terme, plus il concerne ce terme $\rightarrow$ $\text{tf}_{t,d}$ = nombre d'occurrences du terme t dans le document d
- intuition 2 : des termes très fréquents dans tous les documents sont moins importants (moins discriminants) $\rightarrow$ df_t = nombre de documents qui contiennent le terme t
- poids d'un terme $tf.idf_t, d = tf_{t,d} x log_{10} \frac{N}{df_t}$ (N = nombre de documents)

Matrice des poids

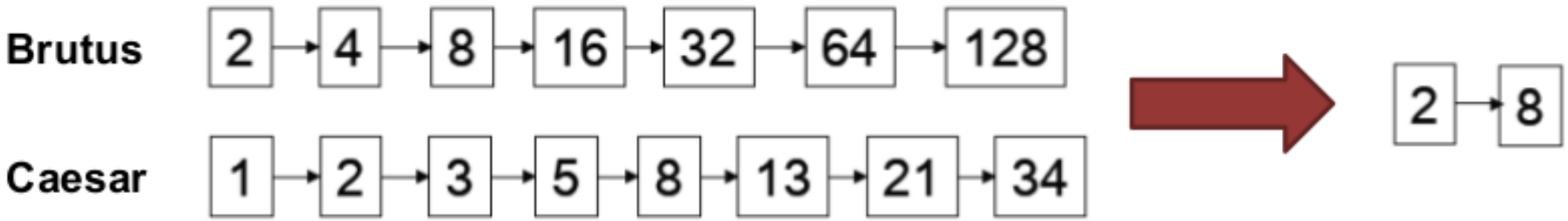
	Antoine & Cléopâtre	Jules César	La Tempête	Hamlet	Othello	Macbeth
Antoine	13,1	11,4	0	0	0	0
Brutus	3,0	8,3	0	1	0	0
César	2,3	2,3	0	0,5	0,3	0,3
Calpurnia	0	11,2	0	0	0	0
Cléopâtre	17,7	0	0	0	0	0
pitié	0,5	0	0,7	0,9	0,9	0,3
pire	1,2	0	0,6	0,6	0,6	0

chaque document est un vecteur dans $\mathbb{R}^{|V|}$

Retrouver les documents

Brutus AND Caesar

- On **recherche** « **Brutus** » dans le dictionnaire
  → On récupère la liste de documents
- On **recherche** « **Caesar** » dans le dictionnaire
  → On récupère la liste de documents
- On **fusionne** les deux listes



Requêtes comprenant des expressions

- Si l'utilisateur formule une requête "Paris Saclay", un document comprenant "Le maire de Paris s'est arrêté dans un restaurant de Saclay aujourd'hui" est probablement non pertinent
- concept d'expression facilement compris par les utilisateurs
- part importante des requêtes web
- index simple insuffisant
- deux solutions :
  - index de n-grammes
  - index positionnel

Notion de n-gramme

- n-gramme = sous-séquence de n éléments extraite d'une séquence donnée
- ici, n-grammes de mots
  - unigrammes : tous les mots
  - bigrammes : séquences de 2 mots etc.
- différent d'un groupe de mots d'un point de vue linguistique

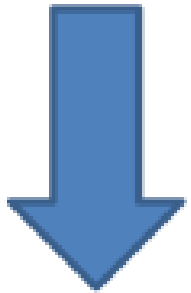
Index de bigrammes

- indexer, en plus des mots simples, les bigrammes du textes
- 1 bigramme = 1 terme du dictionnaire
- en fait, rarement utilisés
  - n-grammes de la requête difficiles à déterminer (Stanford University Palo Alto, Université Paris-Saclay Orsay)
  - vocabulaire de l'index très important

Index de position

- Idée : dans les listes de documents de l'index, ajouter la position de chaque occurrence de terme dans le document

terme	fréquence	→	D1	D3	D4



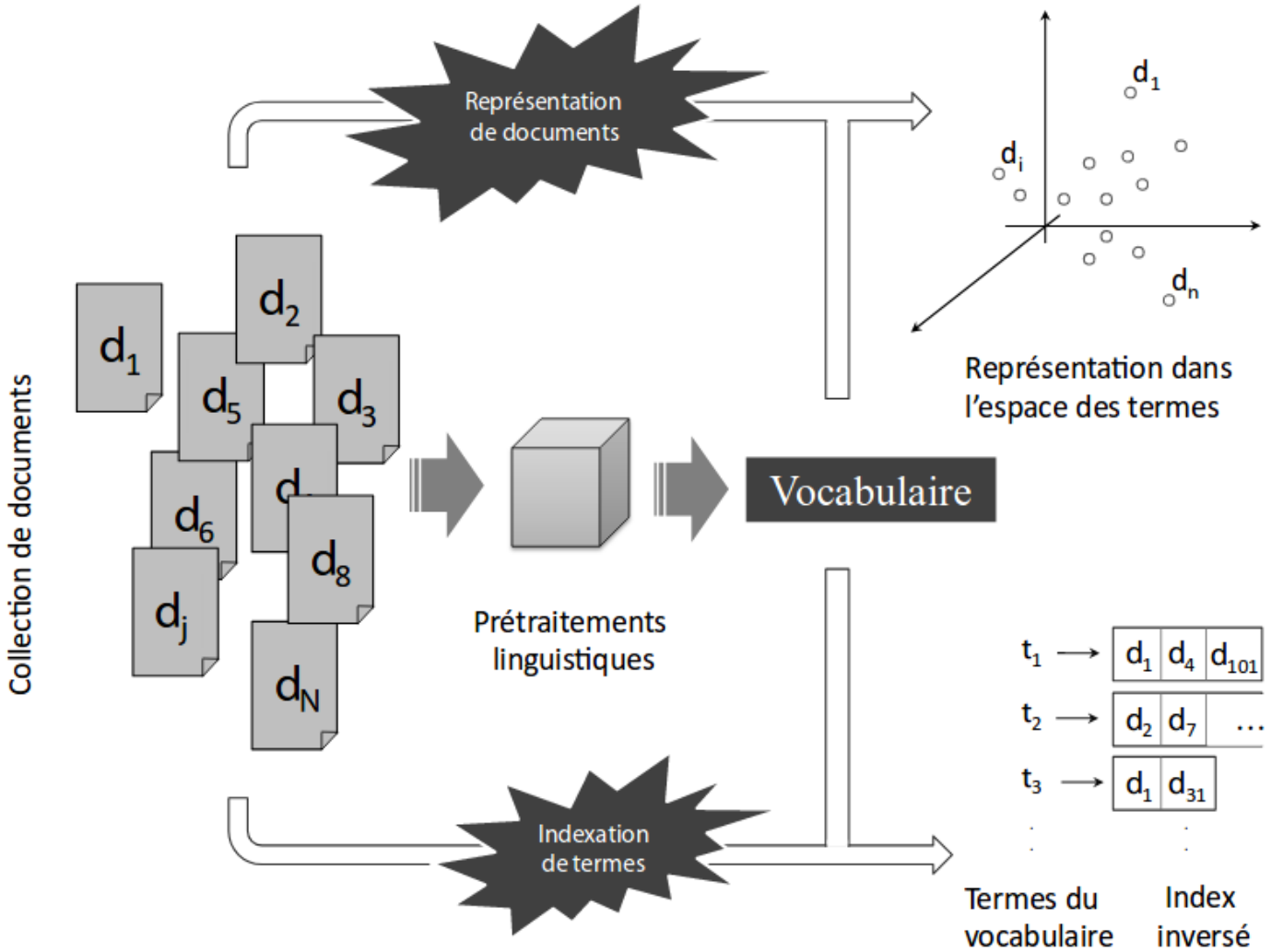
terme	fréquence	→	D1 : pos1, pos2, pos3
			D3 : pos1, pos2
			D4 : pos1, pos2, pos3

Parcours d'un index de position

- "Université Paris Saclay"
  - extraction des entrées du dictionnaire
  - utilisation récursive de l'algorithme de fusion, pour les documents puis pour les positions
  - utilisation d'une comparaison incrémentale au lieu d'une égalité stricte

université	1252	→	D2 : 546
			D6 : 34, **87**, 145, 243
			D7 : 44, 87, 34
			...
paris	45	→	D2 : 547
			D6 : **88**, 543
			...
saclay	15345	→	D2 : 54, 90
			D6 : **89**
			D4 : 43

Plan



Recherche d'information, Applications, modèles et algorithmes de Massih-Reza Amini et Éric Gaussier

Modèles de recherche : les trois courants

- modèles fondés sur la théorie des ensembles
  - modèle booléen
- modèles algébriques
  - modèle vectoriel
- modèles probabilistes
  - modélisation de la notion de pertinence

courants fondés à l'aube de la discipline (60s, 70s)
passage à l'échelle : des bases documentaires jouets au téraoctet de TREC et au web

Modèle booléen

- premier et plus simple des modèles
- fondé sur théorie des ensembles et algèbre de Boole
- termes de la requête soit présents soit absents : poids binaires des termse, 0 ou 1
- document soit pertinent soit non pertinent : pertinence binaire (modèle exact)
- requête exprimée avec opérateurs logiques : AND, OR, NOT
  - (cyclisme OR natation) AND NOT dopage
  - document pertinent ssi son contenu respecte la formule logique demandée

Modèle booléen : exemple

Requête Q : `(cyclisme OR natation) AND NOT dopage`

Le document contient					Pertinence du document
cyclisme	natation	**cyclisme OR natation**	dopage	**NOT dopage**	
0	0	**0**	0	**1**	**0**
0	0	**0**	1	**0**	**0**
0	1	**1**	0	**1**	**1**
0	1	**1**	1	**0**	**0**
1	0	**1**	0	**1**	**1**
1	0	**1**	1	**0**	**0**
1	1	**1**	0	**1**	**1**
1	1	**1**	1	**0**	**0**

Modèle booléen : avantages et inconvénients

Avantages

- précis : document contient termes ou non
- interprétable
- encore utilisé dans nombreux outils, comme messagerie électronique
- adapté pour spécialistes quand vocabulaire contraint préféré (droit)

Inconvénients

- difficile d'exprimer des requêtes longues sous forme booléenne
- critère binaire peu efficace
  - souvent trop ou trop peu de résultats
  - pondération des termes améliore résultats (cf. modèle booléen étendu)
- impossible d'ordonner les résultats
  - tous les documents retournés sont sur le même plan
  - l'utilisateur préfère un classement lorsque la liste est grande

Vers des listes ordonnées de résultats

Pourquoi ordonner les résultats ?

- la plupart des utilisateurs
  - a du mal à écrire des requêtes booléennes
  - ne veut pas parcourir trop de résultats (millions possible)
- préfère des listes ordonnées
  - du plus utile à l'utilisateur (pertinent) au moins utile
  - le nombre de résultats n'est plus un problème
  - l'utilisateur en parcourt autant qu'il le souhaite
- mais nécessite un algorithme d'ordonnancement efficace
- modèle statistique
  - aspect quantitatif des termes et des documents
  - degré de similarité entre une requête et un document

Modèles ordonnancés

Ordonnancement

- le grand nombre de résultats n'est plus un problème : 10 1ers
- suppose que l'algorithme d'ordonnancement fonctionne bien

Principe

- attribuer un score à chaque paire requête-document
  - en fonction de pertinence du document par rapport à la requête
  - généralement présence des termes de la requête dans le document
- trier les documents par score décroissant

Modèle vectoriel

Modèle vectoriel

- Mesure de similarité : représentations proches $\Rightarrow$ probabilité élevée que même information
- Documents et requête représentés par des vecteurs dans un espace euclidien à n dimensions (n : nombre de termes)
  - termes = axes
  - docs = vecteurs (creux)
- Pertinence du document = degré de similarité entre le vecteur de la requête et celui du document
- Documents ordonnés du plus similaire à la requête au moins similaire

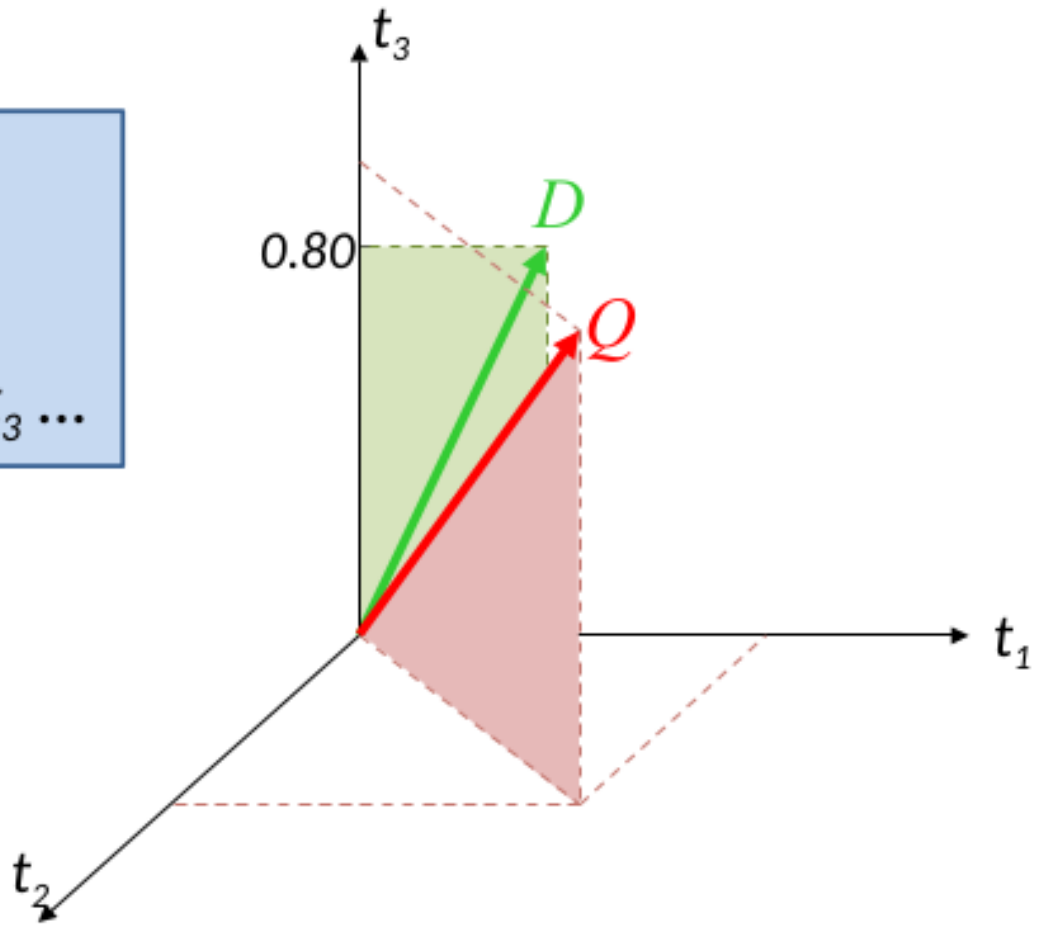
Modèle vectoriel

Requête Q : $t_1\ t_2\ t_3$

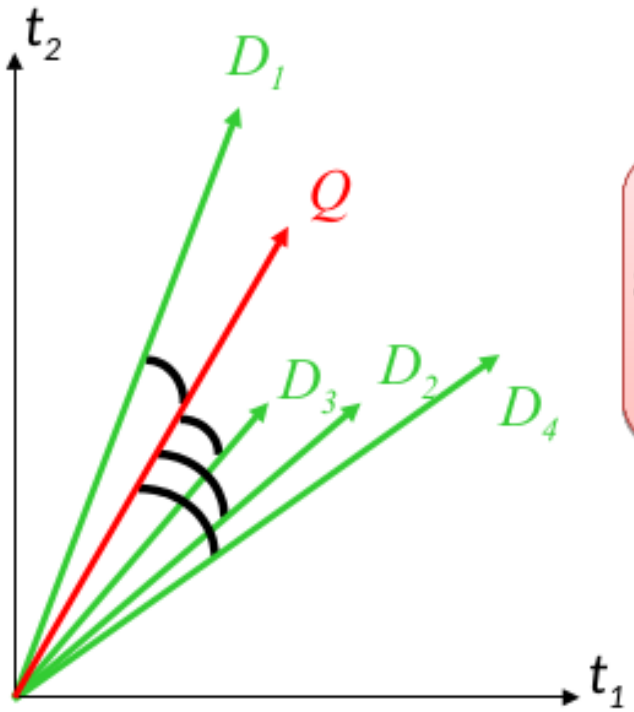
Document D : ... t_1 ... t_3 ...

Poids $w_{D,t1} = 0.45$

Poids $w_{D,t3} = 0.80$



Mesure de similarité



Cosinus

$$\text{sim}(\vec{Q},\vec{D}) = \frac{\vec{Q}\cdot\vec{D}}{|\vec{Q}|\times|\vec{D}|} = \frac{\sum_{i=1}^{n} w_{i,Q}\times w_{i,D}}{\sqrt{\sum w^2{}_{i,Q}}\times\sqrt{\sum w^2{}_{i,D}}}$$

(Le produit scalaire avec normalisation de la longueur des vecteurs)

Modèle vectoriel : avantages et inconvénients

Avantages

- langage de requête simple : liste de mots clés
- performances meilleures que booléen grâce à pondération des termes
- pertinence partielle de documents possible
- tri des documents possible

Inconvénients

- termes considérés comme indépendants
- langage de requête moins expressif
- interprétabilité moindre

Modèle probabiliste

Modélisation du problème

- estimation de la probabilité de pertinence d'un document par rapport à une requête
- notion binaire de pertinence :
  - $R_{d,q} = 1$ si d est pertinent pour q
  - $R_{d,q} = 0$ sinon
- documents ordonnés par probabilité de pertinence décroissance
- pertinence de chaque document supposée indépendante
- dyssymétrie entre requête et document ($\neq$ vectoriel)

Modèle probabiliste : conclusion

- modèle phare :
  - Okapi BM25
    - modèle probabiliste non binaire (fréquence des termes) avec normalisation de la longueur des documents
    - robuste et très utilisé
- autres modèles de type probabiliste
  - réseaux bayésiens
  - modèle de langage
    - document = modèle génératif qui génère la requête
- conclusion
  - problème des probabilités initiales
  - termes indépendants
  - résultats comparables à ceux du modèle vectoriel

Learning to rank

Principe de base

- caractéristiques qui influencent la pertinence
  - requête : longueur, moyenne des idf...
  - document : PageRank, degré d'indésirabilité, date du document...
  - requête + document : similarité cosinus requête document, fenêtre minimale dans laquelle apparaissent les termes de la requête, zones...
- classification binaire : pertinent 1, non pertinent 0
  - mais problème d'apprendre le score de régression
- apprentissage par paire (de documents)
  - mais erreurs n'ont pas toutes le même poids
  - sensible au nombre de docs pertinents par requête
  - premiers documents retournés plus importants
- apprentissage de listes : optimiser MAP directement par exemple
  - problème d'apprentissage difficile

Autres modèles

- modèle vectoriel généralisé
  - représente les dépendances entre termes
  - théoriquement intéressant, mais efficacité non démontrée
- Latent Semantic Indexing
  - propose d'étudier les "concepts" plutôt que les termes (idées d'un texte)
  - lie documents entre eux et avec la requête
  - permet de renvoyer des documents ne contenant aucun mot de la requête
  - moins de dimensions
  - meilleur rappel, moins bonne précision

Plan

Qu'est-ce qu'un bon moteur de recherche ?

Critères

- critère principal : satisfaction utilisateur ?
- rapide
  - analyse rapide de la requête
  - recherche rapide dans l'index
  - tri rapide des résultats
- complet et à jour
  - tous les (ou de nombreux) documents de la collection sont traités
  - nouveaux documents intégrés rapidement

  $\Rightarrow$ construction rapide de l'index

  $\Rightarrow$ (sur le web) découverte permanente, efficace et rapide des nouveaux documents
- le plus important : pertinent

Comment mesurer la pertinence ?

- moteur de recherche sur le web
  - l'utilisateur clique sur certains liens et pas d'autres
  - l'utilisateur retourne sur le moteur
  - l'utilisateur effectue une certaine tâche
- site e-commerce
  - l'utilisateur achète
  - il achète vite
  - une forte proportion de visiteurs achètent
- site d'entreprise
  - l'utilisateur gagne en productivité
  - accès sécurisé
  - ...

Qu'est-ce qu'une bonne évaluation ?

- évaluer un système sert à :
  - savoir s'il remplit la tâche assignée
  - savoir s'il est meilleur que la concurrence
  - savoir comment l'améliorer
- il faut donc une évaluation
  - reproductible
    - pour évaluer plusieurs systèmes de la même façon
    - pour estimer les progrès accomplis
  - interprétable
    - pour identifier les zones de progrès possible
  - rapide
    - pour pouvoir évaluer chaque modification du système indépendamment
  - objective

Comment rendre la pertinence objective ?

- besoin de l'utilisateur transformé en requête → 1re perte d'information
  - besoin d'information : je voudrais savoir si boire du vin rouge réduit le risque de problèmes de coeur
  - requête : vin rouge problèmes coeur
  - doc : le coeur de son discours concernait le problème de l'industrie du vin qui peine à reconnaître le rôle de la consommation de vin rouge dans les accidents de voiture

  → doc pertinent par rapport à la requête mais pas par rapport au besoin
- pertinence des résultats par rapport au besoin d'information initial cependant
- pertinence pas binaire : très pertinent, pas du tout, un peu, pourquoi pas...
- pour rendre pertinence objective, définition simplifiée :
  - documents traités indépendamment les uns des autres
  - pertinence transformée en notion binaire
- et utilisation de collections de test

Méthodologie standard

- collection de documents
  - représentative des documents réels
- ensemble de besoins d'information/requêtes
  - également représentative
- score de pertinence de chaque document pour chaque requête
  - jugements humains

benchmarks standards : TREC

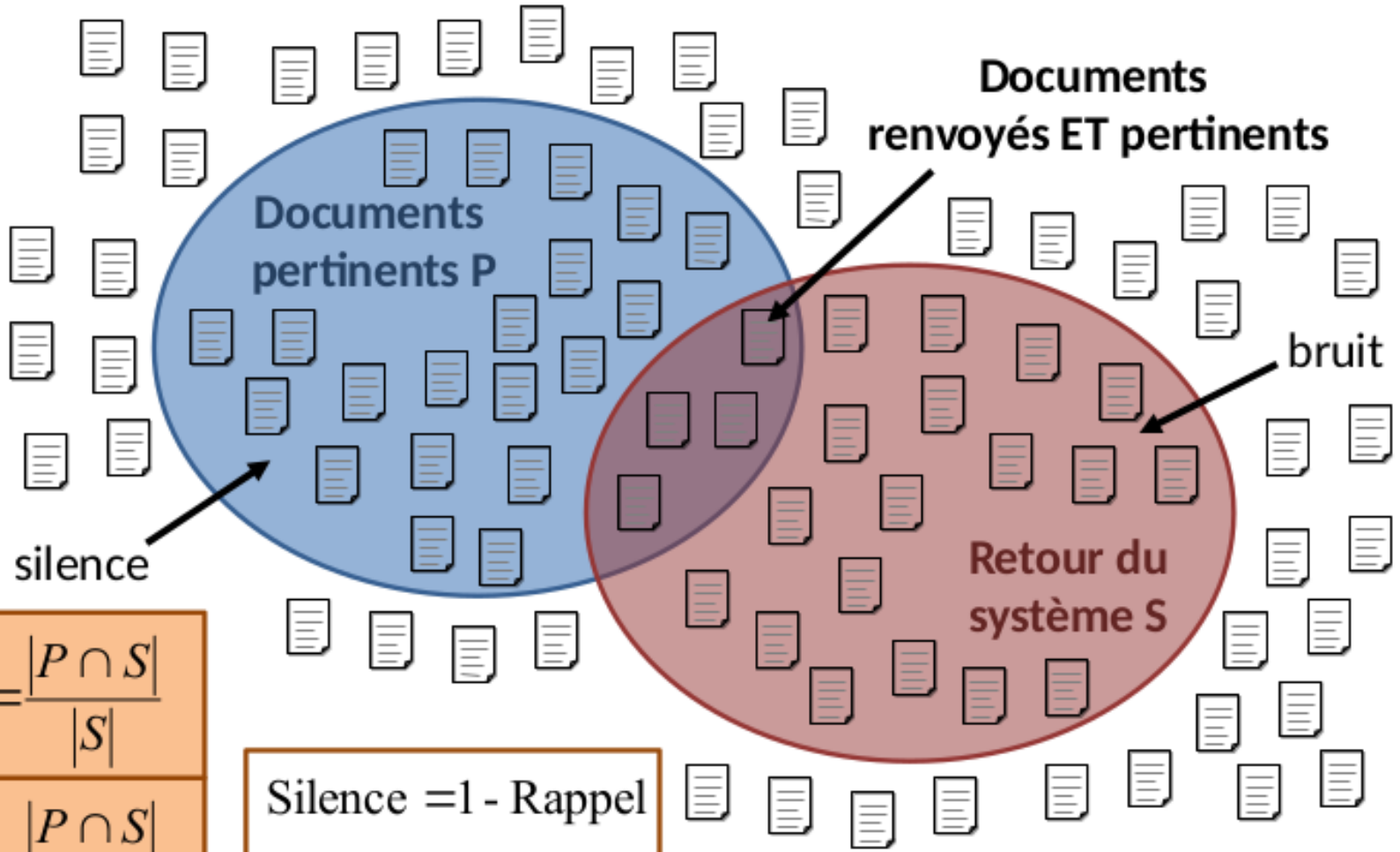
- Ad Hoc en particulier (1992 à 1999)
- scores pour les k premiers documents retournés par les systèmes

Collections de test

La collection de test rend les expériences reproductibles

- On met au point un protocole
- On juge manuellement un nombre significatif d'exemples
  - gold standard
  - Une partie peut également servir d'ensemble de développement et/ou d'apprentissage
- On calcule un accord inter-annotateurs
  - Pour valider le caractère objectif
- On compare les résultats du système aux résultats attendus
- On définit des mesures imparfaites mais précises

Évaluation : précision et rappel



$$\text{Précision} = \frac{|P \cap S|}{|S|}$$

$$\text{Rappel} = \frac{|P \cap S|}{|P|}$$

Silence = 1 - Rappel

Bruit = 1 - Précision

Complémentarité de précision et rappel

Pourquoi pas juste la précision ?

- précision = capacité d'un système à renvoyer SURTOUT des documents pertinents
- Renvoyer un seul document pertinent suffit à obtenir 100% de précision
- pas compatible avec la satisfaction de l'utilisateur !

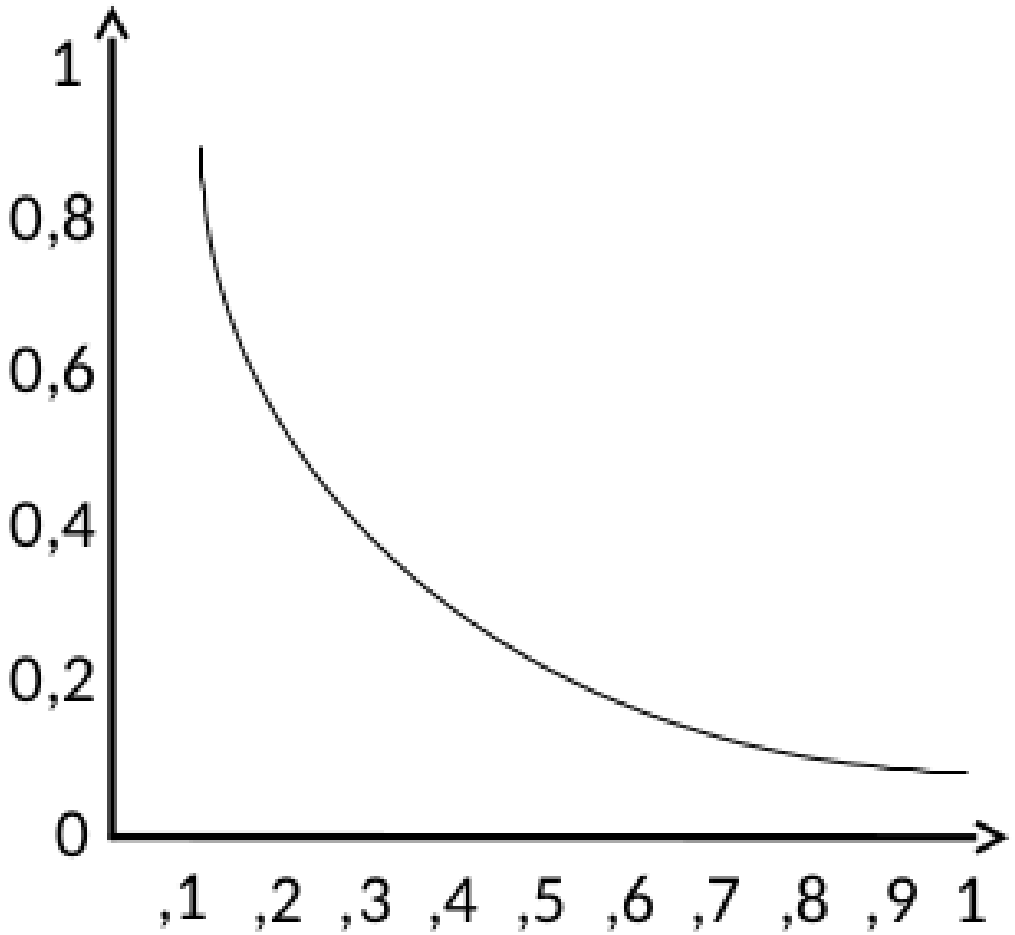
Pourquoi pas juste le rappel ?

- rappel = capacité d'un système à renvoyer TOUS les documents pertinents
- Renvoyer tous les documents de la collection permet d'obtenir 100% de rappel
- pas compatible avec la satisfaction de l'utilisateur !

Courbe rappel/précision

- rappel augmente avec nombre de réponses
- précision diminue

courbe rappel/précision utilisée pour caractériser les systèmes de RI



F-mesure

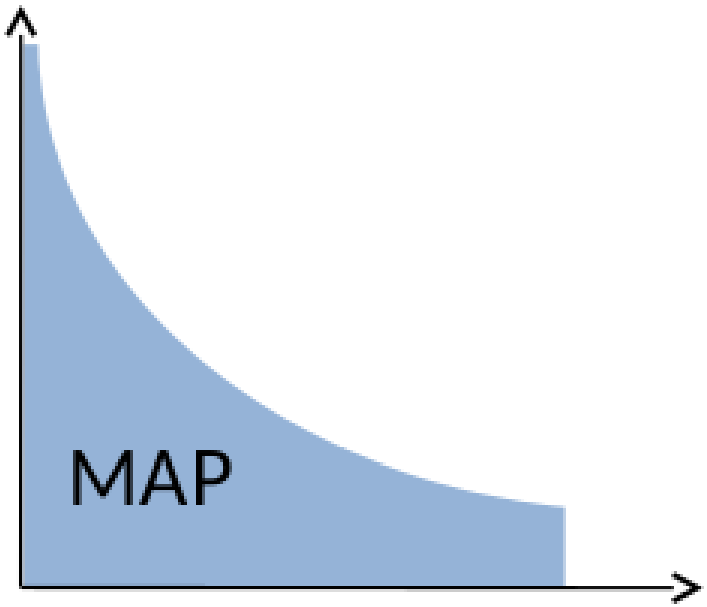
pour obtenir valeur unique, F-mesure = moyenne harmonique
$F = \frac{1}{\alpha\frac{1}{p}+(1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)xPxR}{\beta^2 P+R}$ avec $\alpha = \frac{1}{\beta^2+1}$

$\beta < 1$ favorise la précision, $\beta > 1$ le rappel

pour donner autant d'importance à la précision qu'au rappel, on choit $\beta = 1$
$F = \frac{2PR}{P+R}$

Autres métriques

- MAP (Mean Average Precision) : aire sous la courbe R/P
- pour tenir compte de l'ordonnancement des résultats :
  - P@5, P@10 : précision avec très documents retrouvés ; favorise la haute/très haute précision
  - P@100
  - courbe Rappel/Précision pour k variable
- taux d'erreur = (faux positifs + faux négatifs) / pertinents
- et nombreuses autres...



Présentation des résultats

- l'utilisateur doit pouvoir identifier les documents susceptibles d'être pertinents par leur description $\rightarrow$ titre, url, métadonnées
- souvent résumé en plus
  - statique (ne dépend pas de la requête)
  - dynamique : "snippets" du document qui contiennent les termes

Documents de référence

- Recherche d'information, Applications, modèles et algorithmes de Massih-Reza Amini et Éric Gaussier (2e édition en 2017)
- Introduction to Information Retrieval, Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze (2008)