RICM 4

HMUL8R6B: Accès et recherche d'information.

2024-2025

Modèles de recherche & mesures d'évaluation

Philippe Mulhem & Massih-Reza Amini

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

Plan

- Modèles standard de recherche
- Evaluation

Les différents modèles standard

- Modèle booléen
- Modèle vectoriel
- Modèles probabilistes (non-vus ici)

Notations

x_t^q	Nbre occurrences du terme t dans q
x_t^d	Nbre occurrences du terme t dans le document d
n_t^d	Version normalisée de x_t^d (poids)
N	Nbre de documents dans la collection
M	Nbre de termes dans la collection
F_t	Nbre d'occ. total de t : $F_t = \sum_d x_t^d$
N_t	Fréquence documentaire de t : $N_t = \sum_d I(x_t^d > 0)$
y_d	Longueur du document d
m	Longueur moyenne dans la collection
L	Longueur de la collection
RSV	Retrieval Status Value (score)

Le modèle booléen (1)

Modèle simple fondé sur la théorie des ensembles et l'algèbre de Boole, caractérisé par :

- Des poids binaires (présence/absence)
- Des requêtes qui sont des expressions booléennes
- Une pertinence binaire
- Pertinence système : satisfaction de la requête booléenne

Le modèle booléen (2)

Exemple

$q =$ programmation $\wedge$ langage $\wedge$ (C $\vee$ java)

(dnf : $q =$ [prog. $\wedge$ lang. $\wedge$ C] $\vee$ [prog. $\wedge$ lang. $\wedge$ java])

$q_{c1} = \{"prog.","lang.","C"\}$, $q_{c2} = \{"prog.","lang.","java"\}$

n_t^d (x_t^d)	programmation	langage	C	java	$\cdots$
d_1	1 (3)	1 (2)	1 (4)	0 (0)	$\cdots$
d_2	1 (5)	1 (1)	0 (0)	0 (0)	$\cdots$
d_0	0 (0)	0 (0)	0 (0)	1 (3)	$\cdots$

Score de pertinence

$RSV(d_j, q) = 1$ si $\exists\, q_{cc} \in q_{dnf}$ tq $\forall t, n_t^d = n_t^{qcc}$; 0 sinon

Le modèle booléen (3)

Considérations algorithmiques

Quand la matrice documents-termes est creuse (lignes et colonnes), utiliser un fichier inverse pour sélectionner le sous-ensemble des documents qui ont un score de pertinence non nul avec la requête (sélection rapidement réalisée). Le score de pertinence n'est alors calculé que sur les documents de ce sous-ensemble (généralisation à d'autres types de score).

	d_1	d_2	d_3	$\cdots$
programmation	1	1	0	$\cdots$
langage	1	1	0	$\cdots$
C	1	0	0	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	

Le modèle booléen (4)

Avantages et désavantages

\+ Facile à développer

\- Pertinence binaire ne permet pas de tenir compte des recouvrements thématiques partiels

\- Passage d'une besoin d'information à une expression booléenne

Remarque À la base de beaucoup de systèmes commerciaux

Le modèle vectoriel (1)

Revient sur deux défauts majeurs du modèle booléen : des poids et une pertinence binaires
Il est caractérisé par :

- Des poids positifs pour chaque terme dans chaque document
- Mais aussi des poids positifs pour les termes de la requête
- Une représentation vectorielle des documents et des requêtes

Le modèle vectoriel (2)

On considère donc que les documents et les requêtes sont des vecteurs dans un espace vectoriel de dimension M dont les axes correspondent aux termes de la collection

Similarité Cosinus de l'angle entre les deux vecteurs

$$RSV(d,q) = \frac{\sum_t n_t^d n_t^q}{\sqrt{\sum_t (n_t^d)^2}\sqrt{\sum_t (n_t^q)^2}} \text{ on a } n_t^d = w_{id}$$

Propriété Le cosinus est maximal lorsque document et requête contiennent exactement les mêmes termes, dans les mêmes proportions ; minimal lorsqu'ils n'ont aucun terme en commun (*degré de similarité*)

Le modèle vectoriel (3)

Calcul des poids d'un terme t :

$\rightarrow$ Pour les termes du document : $n_t^d = w_{id}$

$\rightarrow$ Pour les termes de la requête : $n_t^q = tf_{t_i,q} \times idf_{t_i}$
avec idf_{t_i} calculé sur le corpus.

Le modèle vectoriel (4)

Traitement de requête avec index inversé :

- Requête q = $\{n_{t1}^q ... n_{tM}^q\}$
- On garde les *ti* tels que $n_{ti}^q \neq 0$
- Boucle sur ces *ti* et sur les *dj* (indice j sur les docs) :
  - ligne_res[j] += $n_{ti}^{dj} * n_{ti}^q$
    (par utilisation des lignes de l'index inversé)
- Calcul final : ligne_res[j] = $ligne\_res[j]/(||dj|| * ||q||)$
- Tri des résultats par ordre décroissant, filtrage et affichage

Le modèle vectoriel (5)

Avantages et désavantages

\+ Schémas de pondération permettant de prendre en compte différentes propriétés des index

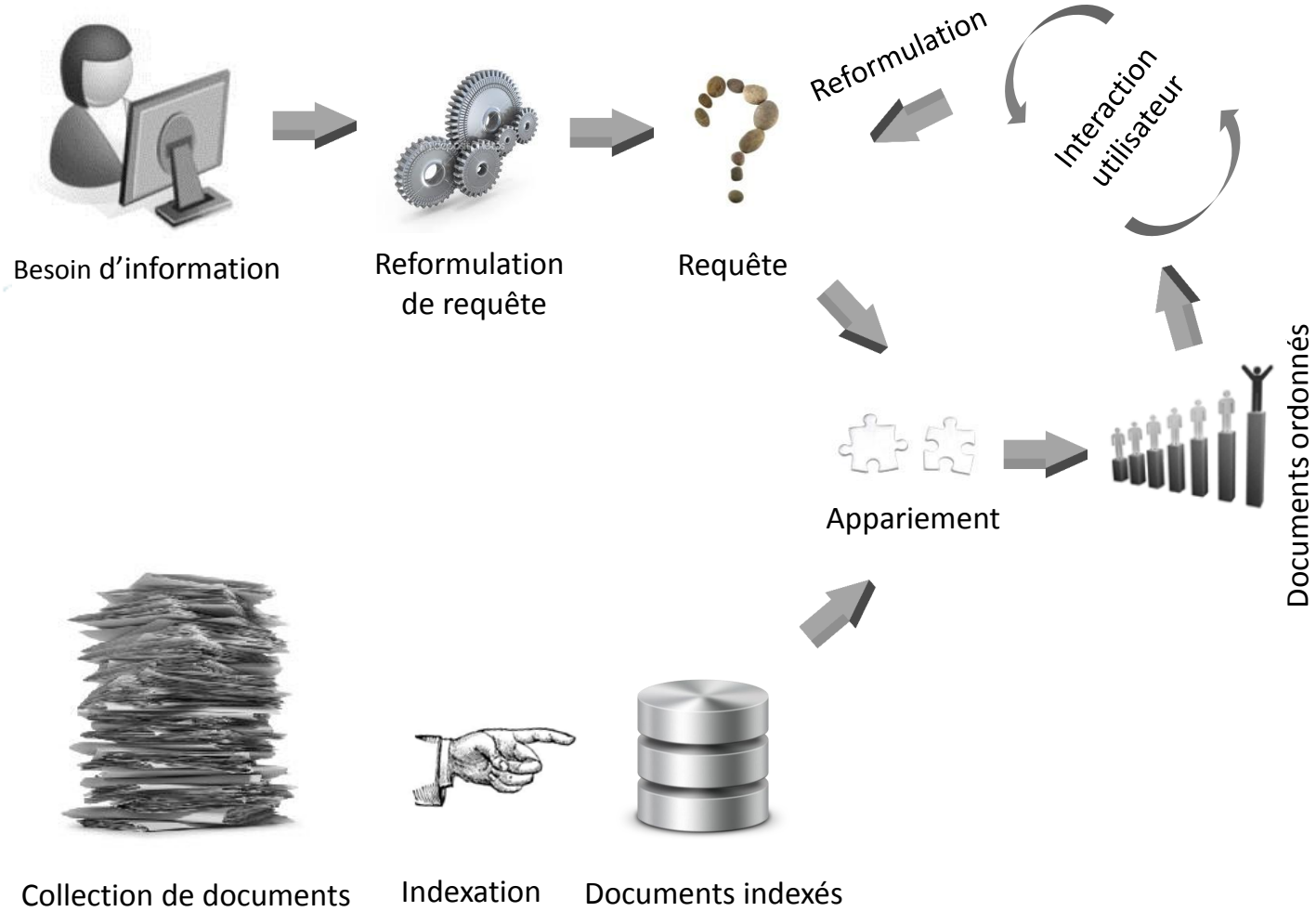
\+ Un appariement partiel qui permet de retrouver les documents qui répondent en partie à la requête

\+ Un ordre total sur les documents qui permet de distinguer les documents qui abordent pleinement les thèmes de la requête de ceux qui ne les abordent que marginalement

\- Difficulté d'aller plus avant dans le cadre vectoriel (modèle relativement simple)

Complexité : comme le modèle booléen, linéaire sur le nombre de documents qui contiennent les termes de la requête (similarité requête-document plus coûteuse)

Expansion de requêtes



Algorithme de Rocchio

- La formule de Rocchio consiste alors à enrichir la requête initiale q_0 avec les termes de la requête $q^\star$, cet enrichissement étant contrôlé par des poids qui peuvent être réglés automatiquement (ou manuellement) sur de nouvelles collections:

$$\mathbf{q}^{\text{new}} = \alpha \mathbf{q}_0 + \beta \frac{1}{||\mathcal{D}_p||} \sum_{d \in \mathcal{D}_p} \mathbf{d} - \gamma \frac{1}{||\mathcal{D}_{np}||} \sum_{d' \in \mathcal{D}_{np}} \mathbf{d}'$$

  avec α, β et γ des réels positifs ou nuls (typiquement 1, 0.6, 0.4).
- On utilise aussi les cas avec uniquement des documents positif, sans la dernière partie de la formule avec γ.

Evaluations

Un élément fondamental en recherche d'information : comment évaluer la qualité d'un système

Les jugements/annotations les plus fréquents

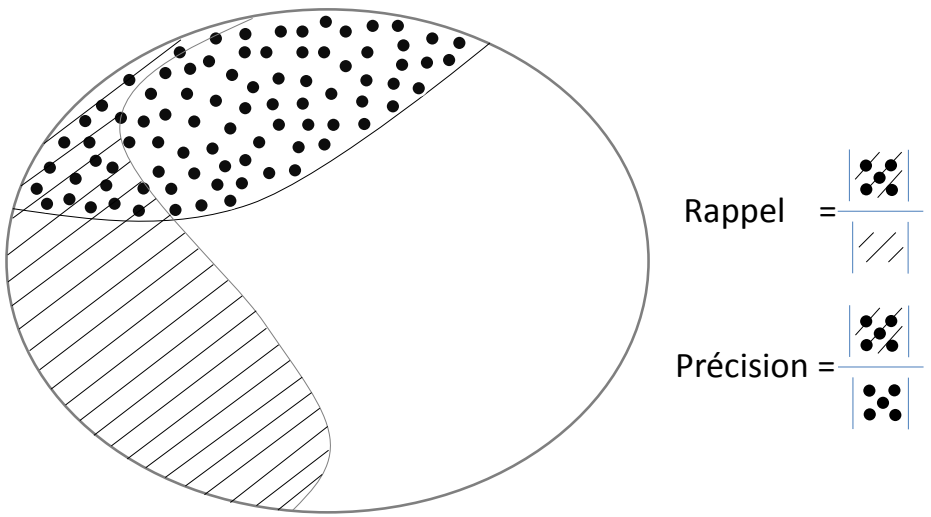
- Jugements binaires : ce document est pertinent (1) ou non (0) pour cette requête
- Jugements multi-valués :
  $Parfait > Excellent > Bon > Correct > Mauvais$
- Paires de préférence : document d_A plus pertinent que document d_B pour cette requête

Mesures d'évaluations, jugements binaires

Les deux mesures d'évaluation les plus utilisées en RI sont le *rappel* et la *précision*:

$$\text{Rappel} = \frac{\text{Nbre de documents pertinents retournés par le système}}{\text{Nbre de documents pertinents}}$$

$$\text{Precision} = \frac{\text{Nbre de documents pertinents retournés par le système}}{\text{Nbre de documents retournés}}$$

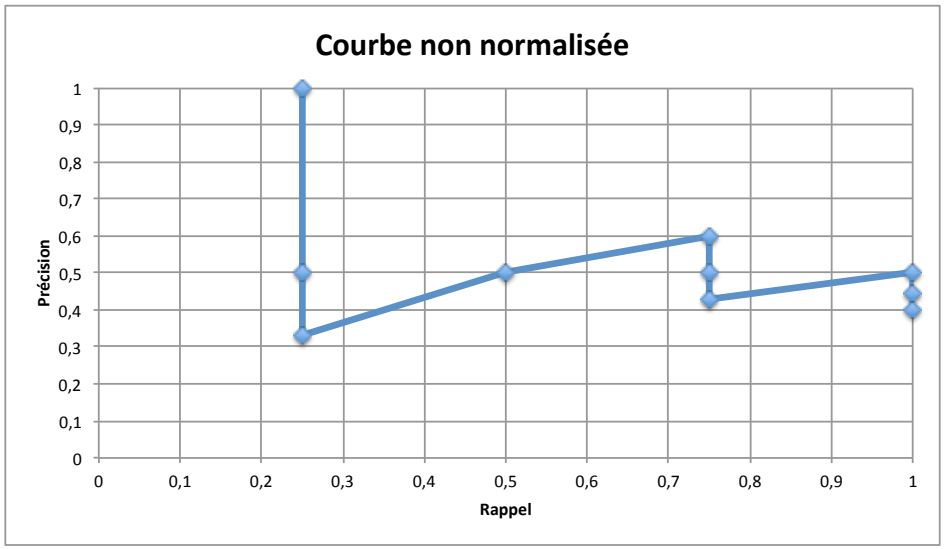


Courbes précision/rappel non-normalisées

rg	Réponse (id : score)	$R_{d_{rg},q}$	rappel, r	précision, p
1	d_{12} : 0.95	1	1/4	1
2	d_4 : 0.82	0	1/4	1/2
3	d_{74} : 0.75	0	1/4	1/3
4	d_{239} : 0.7	1	1/2	1/2
5	d_{38} : 0.65	1	3/4	3/5
6	d_{42} : 0.5	0	3/4	1/2
7	d_1 : 0.4	0	3/4	3/7
8	d_{98} : 0.35	1	1	1/2
9	d_{76} : 0.2	0	1	4/9
10	d_{74} : 0.1	0	1	2/5

Table: *rappel* et de *précision* sur un ensemble de 10 documents en réponse d'un moteur de recherche $\mathcal{M}$ pour une requête q ayant 4 documents pertinents : d_{12}, d_{239}, d_{38}, d_{98}.

Courbes Précision/Rappel non-normalisées

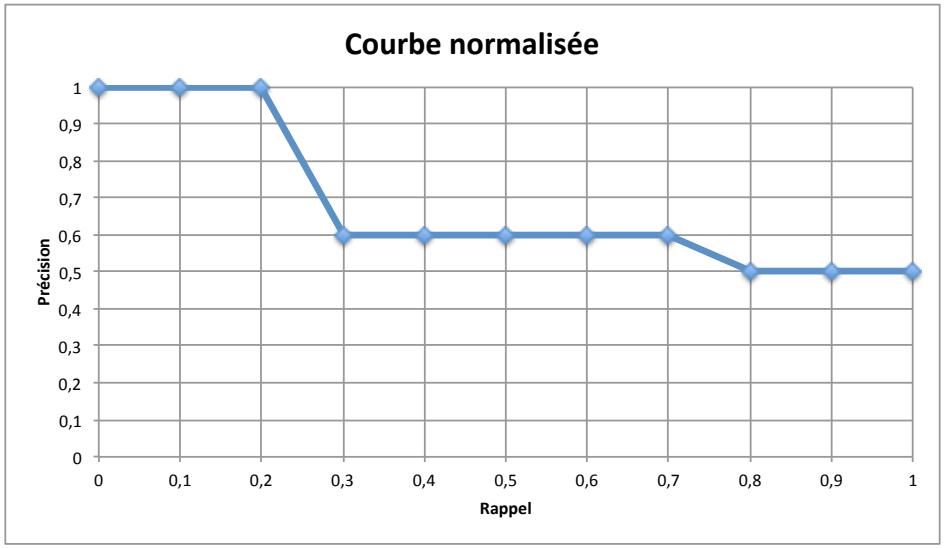


Courbes précision/rappel normalisées

rappel, r	précision, $\mathtt{P}(r)$
0	1
0.1	1
0.2	1
0.3	3/5
0.4	3/5
0.5	3/5
0.6	3/5
0.7	3/5
0.8	1/2
0.9	1/2
1	1/2

Table: Tableau normalisé : 1. on fixe R dans 0, 0,1, ... 0,9, 1 ; 2. on sélectionne les lignes non-normalisées avec r$\geq$R ; 3. on sélectionne la P(r) max.

Courbes Précision/Rappel normalisées



Courbes Précision/Rappel normalisées

Pour évaluer un système

- Un ensemble de requêtes q_j (50+)
- On fait la moyenne, pour toutes les requêtes, des tableaux normalisés, valeur de rappel par valeur de rappel, pour obtenir un seul tableau normalisé qui synthétise la qualité du système.

D'autres mesures classiques

- La précision moyenne(*Average Precision* en anglais) d'un système de recherche $\mathcal{M}$ pour une requête q donnée, notée souvent *AveP*, est la moyenne des valeurs de précision des documents pertinents par rapport à q dans la liste ordonnée des réponses:

$$AveP(q) = \frac{1}{n_+^q} \sum_{k=1}^{N} R_{d_k,q} \times \mathtt{P@}k(q)$$

où n_+^q est le nombre total de documents pertinents par rapport à q. Dans notre exemple AveP(q)=0,65.

- Mean Average Precision

$$MAP = \frac{1}{|Q|} \sum_{j=1}^{|Q|} AveP(q_j)$$