Master SRI
(Systèmes de recherche d'informations)

RECHERCHE D'INFORMATION TEXTUELLE

http://larbiguezouli.com
ou
http://fac-sciences.univ-batna.dz/cs/enseignants/guezouli_larbi_site/index.html

Présenté par Dr Larbi GUEZOULI

Sommaire

Sommaire
(suite)

Chapitre I
Indexation

I. Introduction à la recherche d'information

1. Définitions

- Un système de recherche d'information (SRI) permet de **retrouver** les documents **pertinents** à une **requête** d'utilisateur, à partir d'une grande base de documents.
- Un **document** peut être un **texte**, un **morceau** de texte, une **page Web**, une **image**, une **vidéo**, etc.
- Une **requête** exprime le **besoin** d'information d'un utilisateur.
- La **pertinence**: Dans un document **pertinent**, l'utilisateur doit **trouver** les informations dont il a besoin.

Exemple: Un livre peut être spécifié comme suit:

ISBN: 978-0201398298
Auteur: Ricardo Baeza-Yates, Berthier Ribeiro-Neto.
Titre: Modern Information Retrieval
Editeur: Addison-Wesley
Date: 1999
…
Contenu: <Texte du livre>

Exemple: (suite)

Il est possible de chercher ce livre par les **attributs** (ISBN, Auteur, …, Date).

On peut aussi chercher ce livre par le **contenu**.

La recherche par attributs est relativement simple. Par contre, la recherche par le contenu est l'objet des études sur la RI.

Approches possibles:

2. Approche naïve

Consiste à balayer les documents **séquentiellement** et les comparer avec la requête. Si on trouve la **même** chaîne de caractère dans un document il sera sélectionné comme **réponse**.

Approche très **simple** mais:

•La recherche est très **lente**;

•La requête est une simple chaine de caractères qui doit être trouvée à **l'exacte**.

3. Approche basée sur une indexation

Des **prétraitements** sur les documents et les requêtes sont indispensables pour **construire** une structure **d'index** qui permet de retrouver très **rapidement** les documents incluant les mots demandés.

La structure d'index la plus utilisée est celle du fichier inversé: mot → {doc_i, …}

Une **requête** peut être une expression complexe incluant des opérateurs logiques (ET, OU, …) ou d'autres types d'opérateurs.

3. Approche basée sur une indexation (suite)

Avantages:

- Plus **rapide**
- L'expression des requêtes permet d'exprimer des besoins d'information **complexes**.

Inconvénients:

- La **structure d'index** exige un espace de stockage **important** (de 40% à 200% de la taille de collection de documents) selon la complexité de l'indexation.

4. La pertinence

Plusieurs définitions:

- La **correspondance** entre un document et une requête
- Degré de **relation** (chevauchement, relativité, …) entre le document et la requête
- Une **mesure** d'utilité du document pour l'utilisateur

4. La pertinence (suite)

Tefko Saracevic propose la définition suivante dans son livre « Introduction to information science », chapitre 3:

- **La pertinence est la A d'un B existant entre un C et un D jugé par un E.**

où A : intervalle de la mesure
B : aspect de la pertinence
C : un document
D : besoin d'information (requête)
E : l'utilisateur

4. La pertinence (suite)

Exemple:

La pertinence est la taille de la partie parlant sur la pédiatrie (A) du domaine de la médecine (B) existant entre le document C et la question D jugé par l'utilisateur E.

II. Indexation de documents

1. Objectif

L'indexation doit permettre de sélectionner des **représentants** du document (mots clés).

Ces **représentants** permettent de **décrire** le contenu (la sémantique) du document et de la requête de façon assez **précise**.

Ils peuvent être des mots composés, par exemple:

Un document représenté par le mot composé «Recherche d'information» ne peut pas être représenté par les deux mots séparés « recherche » et « information ».

2. Approche basée sur la fréquence d'occurrences

Cette approche consiste à **choisir** les mots représentants selon leurs **fréquences d'occurrences**.

Pour cela il faut définir un **seuil minimal** sur la fréquence: si la fréquence d'occurrence d'un mot **dépasse** ce seuil, alors il est considéré comme **important** pour le document.

Cependant, les **statistiques** montrent que les mots les plus fréquents sont des **mots vides**.

Ce phénomène à été **traité** par la **loi de Zipf**:

La loi de Zipf

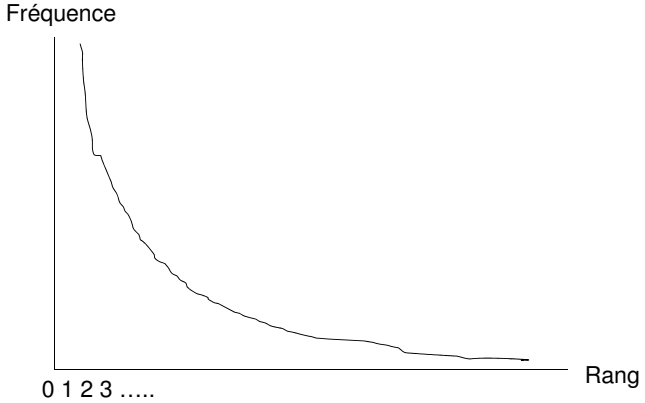
Les mots sont classés dans l'ordre décroissant de leurs fréquences, puis on leur affecte un numéro de rang (1, 2, …), alors:

rang * fréquence ≅ constante

Exp.

Rang	Mot	Fréquence	Rang * Fréquence
1	le	69971	69971
2	de	37411	74822
3	et	25852	77556
4	à	20149	80596
5	un	16237	81185
6	dans	13341	80046
7	que	10595	74165

La loi de Zipf (suite)



La loi de Zipf (suite)

D'après le graphe, il devient évident qu'il ne faut pas garder tous les mots les plus fréquents. Pour cela il faut définir un deuxième **seuil maximal** qui permet d'éliminer les mots de très grande fréquence.

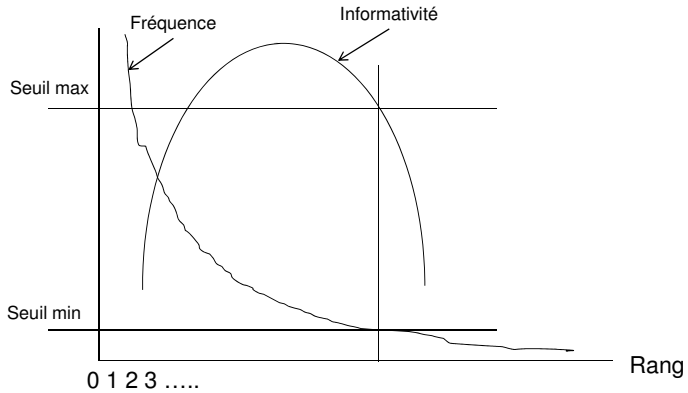
Ce seuil maximal est réglé selon **l'informativité** de mots.

L'informativité est la **quantité** de **sens** que porte un mot.

Cette notion **n'est pas définie** très précisément dans la RI. Elle est utilisée seulement de façon **intuitive**.

La loi de Zipf (suite)

La correspondance entre l'informativité et la fréquence est représentée dans la figure suivante:



La loi de Zipf (suite)

Ainsi, en choisissant les mots qui ont des fréquences **entre les deux seuils**, on espère obtenir les mots dont **l'informativité** est la plus **élevée**.

3. Approche basée sur la valeur de discrimination

Un terme est dit **discriminant** s'il distingue bien un document des autres.

Un terme qui **apparaît** dans **tous** les documents n'est **pas discriminant**.

Le calcul de la valeur de discrimination a été développé dans le modèle vectoriel.

Exp. Soit le document d_i : $<p_{i1}, p_{i2}, ..., p_{in}>$

où p_{ij} est le poids du terme t_j dans le document d_i.

n est le nombre de termes du corpus (Vocabulaire).

3. Approche basée sur la valeur de discrimination (suite)

Le calcul de la valeur de discrimination d'un terme t_k se fait comme suit:

- Calculer le vecteur centroïde **V** du corpus:

Le poids du terme t_j dans **V** est la moyenne de ses poids dans les documents:

$$p_j = \frac{\sum_{i=1}^{N} p_{ij}}{N}$$

tel que: N est le nombre de documents.

3. Approche basée sur la valeur de discrimination (suite)

- Calculer l'uniformité du corpus U_1 (similarité moyenne) comme suit:

$$U_1 = C \times \sum_{i=1}^{N} sim(d_i, V)$$

tel que:

C est une constante de normalisation (souvent = $1/N$)

$sim(d_i, V) \in [0,1]$: la similarité entre le document d_i et **V**.

3. Approche basée sur la valeur de discrimination (suite)

- Uniformiser le poids du terme en question t_k à 0, et répéter les deux étapes précédentes pour calculer U_2.
- Calculer la valeur de discrimination: $v_k = U_2 - U_1$

Exp.

d_1: 6 2 3 6 2 V : 6 2.667 2 2 1.333

d_2: 6 1 2 0 2 V' : 6 2.667 2 0 1.333 ($d_{1,2,3}$[4]=0 pour k=4)

d_3: 6 5 1 0 0 V'': 0 2.667 2 2 1.333 ($d_{1,2,3}$[1]=0 pour k=1)

Chapitre I
Indexation

Si on calcul la similarité comme le complément de la distance euclidienne:

$$sim(d_i, V) = 1 - \sqrt{\frac{\sum_{j=1}^{M} |d_{ij} - V_j|^2}{\text{maxSomme}}}$$

tel que:

M: nombre de termes dans chaque document (= 5)

maxSomme: est la valeur maximale de la somme pour normaliser entre 0 et 1. $\sum_{j=1,i=1}^{j=M,i=N} \max(d_{ij})^2 = M * \max(d_{ij:1\le i\le M, 1\le j\le N})^2$

Chapitre I
Indexation

Calculons maintenant la valeur de l'uniformité U_1:

$$U_1 = C \times \sum_{i=1}^{N} sim(d_i, V)$$

tel que $C = 1/N$ et N=3 documents

$$U_1 = \frac{1}{3} \cdot (sim(d_1 - V) + sim(d_2 - V) + sim(d_3 - V))$$

$$U_2 = \frac{1}{3} \cdot (sim(d_1 - V') + sim(d_2 - V') + sim(d_3 - V')) \text{ pour V'}(t_4)$$

$$U_3 = \frac{1}{3} \cdot (sim(d_1 - V'') + sim(d_2 - V'') + sim(d_3 - V'')) \text{ pour V''}(t_1)$$

Chapitre I
Indexation

D1	6	2	3	6	2			
D2	6	1	2	0	2			
D3	6	5	1	0	0		maxSomme	180,000
V	6,000	2,667	2,000	2,000	1,333			
V1	0,000	2,667	2,000	2,000	1,333			
V2	6,000	0,000	2,000	2,000	1,333			
V3	6,000	2,667	0,000	2,000	1,333			
V4	6,000	2,667	2,000	0,000	1,333			
V5	6,000	2,667	2,000	2,000	0,000			
simD1V	0,685		simD1V1	0,685		simD1V2	0,689	
simD2V	0,800		simD2V1	0,800		simD2V2	0,843	
simD3V	0,739		simD3V1	0,739		simD3V2	0,806	
simD1V3	0,694		simD1V4	0,898		simD1V5	0,689	
simD2V3	0,800		simD2V4	0,866		simD2V5	0,806	
simD3V3	0,750		simD3V4	0,786		simD3V5	0,759	
U	0,741	U1	0,741	U2	0,779	U3	0,748	
U4	0,850	U5	0,751					
v1	0,000	v2	0,038	v3	0,007	v4	0,109	
v5	0,010							

Chapitre I
Indexation

Calculons maintenant la valeur de discrimination pour les deux termes t_4 et t_1:

$$v_4 = U_2 - U_1 = 0.850 - 0.741 = 0.109 \text{ pour V'}(t_4)$$

$$v_1 = U_3 - U_1 = 0.741 - 0.741 = 0 \text{ pour V'}(t_1)$$

Nous observons que t_4 est discriminant puisque il ne se trouve que dans d_1 avec un grand poids.

Alors que t_1 n'est pas discriminant puisque il se trouve avec un grand poids dans tous les documents.

4. Approche basée sur tf*idf

tf*idf est une valeur très utilisée dans le domaine de la RI.

tf (term frequency) : Ce facteur mesure **l'importance** d'un terme pour un document, il est proportionnelle à la **fréquence** du terme dans le document (**pondération locale**). Elle peut être utilisée telle quelle ou selon plusieurs déclinaisons:

- **tf = f(t,d)** fréquence du terme dans un document;
- **tf = f(t,d)/Max[f(t,d)]** où Max[f(t,d)] est la fréquence maximale des termes dans d;
- **tf = log(f(t,d))**
- **tf = log(f(t,d) + 1)**

4. Approche basée sur tf*idf (suite)

idf (inverse of document frequency) : Ce facteur mesure la **discrimination** d'un terme dans le corpus (**pondération globale**).

Un terme qui **apparait souvent** dans le corpus est **moins important** qu'un terme **moins fréquent**.

idf est généralement exprimé comme suit:

$$idf = \log_{10}\left(\frac{N}{df}\right)$$

oú df : est le nombre de documents contenant le terme;

N : est le nombre total de documents du corpus.

4. Approche basée sur tf*idf (suite)

Ainsi, un terme qui a une valeur de **tf*idf élevée** doit être à la fois **important** dans ce document, et aussi il doit **apparaître peu** dans les autres documents.

Exp.

Soit le corpus suivant:

D1:<a,b,c,d,e,c,f,g,c,h>

D2:<i,j,k,l,m,n,o,p,q>

D3:<r,s,t,u,v,w,x,c,y,z>

Calculons la pertinence du terme t_1='c' pour les 3 documents en utilisant tf*idf.

4. Approche basée sur tf*idf (suite)

Exp. (suite)

$tf_{1,1} = 3$;

$tf_{1,2} = 0$;

$tf_{1,3} = 1$;

$idf_1 = \log_{10}(3/2) = 0.1761$

ce qui fait:

$tf_{1,1} * idf_1 = 0{,}5283$

$tf_{1,2} * idf_1 = 0$

$tf_{1,3} * idf_1 = 0.1761$

Donc, le premier document D1 apparaît ainsi comme **le plus pertinent** pour le terme t_1.

Chapitre I
Indexation

5. Filtrage des mots fonctionnels (mots vides)

Sont des mots qui ne portent pas de sens dans le document.

Ces mots sont souvent des prépositions (de, à, …), des prénoms (aucun, tout, on,…), certains adverbes (ailleurs, maintenant, …), et adjectifs (certain, possible,…), etc.

Pour les éliminer on utilise une liste appelée **stoplist** (anti-dictionnaire) qui contient ces mots fonctionnels.

Chapitre I
Indexation

6. Lemmatisation

Les mots qui ont le même sens mais avec des formes différentes doivent être reconnu comme une seule entité lexicale, comme:

transformer, transforme, transforment, transformation, transformateur…

Ces mots ont la même racine (lemme) qui est « transformer »

Il existe plusieurs façon de lemmatisation:

Chapitre I
Indexation

6. Lemmatisation (suite)

a. Une **première méthode** consiste à **examiner** la forme du mot et essayer de **déduire** la racine. Avec cette méthode on doit générer un **algorithme** pour chaque **langue**.

b. Une **deuxième** méthode consiste à utiliser un **dictionnaire** de tous les mots d'une langue avec leurs origines. Cette méthode est plus efficace mais nécessite un dictionnaire qui n'est pas facile à trouver.

c. D'autres méthodes existent comme celle qui utilise un dictionnaire de terminaisons …

Chapitre II
Modélisation

1. Caractérisation formelle des modèles de RI
2. Les modèles classiques de la recherche d'information
   a) Modèle booléen
      Le modèle booléen étendu
   b) Modèle vectoriel
      Quelques mesures de similarités
      Le modèle LSI
   c) Modèle probabiliste
   d) Les modèles basés sur les réseaux de neurones
   e) Conclusion

1. Caractérisation formelle des modèles de RI

Un modèle de recherche d'information est vu comme un quadruplet **[D, Q, F, R(q_i,d_j)]** tel que:

D: est l'ensemble des représentations des documents du corpus;

Q: est l'ensemble des représentations des requêtes de l'utilisateur;

F: est le Framework de modélisation des représentations des documents (ensembles des opérations sur les représentations des documents);

R(q_i,d_j): est la fonction de classement qui associe au couple (q_i,d_j) un réel représentant le degré de rapprochement entre $\mathbf{q_i}$ et $\mathbf{d_j}$.

2. Les modèles classiques de la recherche d'information

a. Modèle booléen

Dans le modèle booléen, les documents et les requêtes sont représentés comme des **ensembles de termes indexes**.

b. Modèle vectoriel

Dans le modèle vectoriel, les documents et les requêtes sont représentés comme des **vecteurs** dans un espace à **n** dimensions.

2. Les modèles classiques de la recherche d'information (suite)

c. Modèle probabiliste

Dans le modèle probabiliste, la modélisation des documents et des requêtes est basée sur la **théorie des probabilités**.

d. Modèle basé sur les réseaux de neurones

Ce modèle est basé sur les réseaux de neurones à deux couches (entrée + sortie).

a. Le modèle booléen

Ce modèle est un simple modèle basé sur la **théorie des ensembles** et **l'algèbre booléen**.

Ce modèle considère qu'un terme est **présent** ou **absent** dans un document, $w_{i,j} \in \{0,1\}$.

Une **requête** est **constituée** d'un ensemble de **termes** séparés par les **opérateur** booléens AND, OR, NOT.

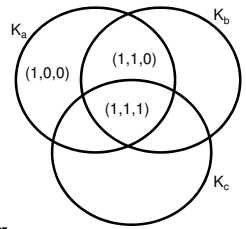
Exp. 1

$r = K_a \cap (K_b \cup \neg K_c)$

r sera vrai dans les 3 cas:

(K_a, K_b, K_c) : { (1,0,0), (1,1,0), (1,1,1) }

Donc un document $\mathbf{d_j}$ qui contient $\mathbf{K_b}$ et ne contient pas $\mathbf{K_a}$ ni $\mathbf{K_c}$ (0,1,0) n'est **pas pertinent** par rapport à la requête **q**.



Chapitre II
Modélisation

a. Le modèle booléen (suite)

Exp. 2

Soit la représentation binaire d'un corpus de 4 documents:

	t_1	t_2	t_3	…	t_n
D1	1	1	0		0
D2	1	0	1		0
D3	1	1	1		1
D4	0	0	1		1

avec la requête **$R = t_1$ AND (t_2 OR t_3) AND NOT t_n**

Chapitre II
Modélisation

a. Le modèle booléen (suite)

Exp. 2 (suite)

Les documents du corpus qui satisfont la requête **R** sont D_1 et D_2.

Remarque:

Le modèle booléen ne **permet pas** de définir la notion de **ressemblance**.

Chapitre II
Modélisation

a. Le modèle booléen (Etendu)

Le modèle **booléen étendu** a été défini en 1983 par Salton, Fox et Wu. C'est une **extension** du modèle booléen qui vise à l'assouplir pour autoriser la sélection de **documents ressemblants** et **organiser** les documents retenus par ordre d'importance.

Les termes des documents sont représentés par des **réels** dans [0, 1] qui représentent le **poids** du terme dans le document.

Les opérateurs booléens AND, OR et NOT sont **étendus** à l'intervalle des réels [0, 1]. Soit (a, b) un couple de réels dans [0,1]x[0,1]:

a AND b = min(a, b)

a OR b = max(a, b)

NOT a = 1 - a

Chapitre II
Modélisation

a. Le modèle booléen (Etendu)

On définit donc des **opérateurs normés**. Les opérateurs **n-AND, n-OR et NOT** sont définis dans le tableau suivant.

Le coefficient **n** est un réel entre 1 et ∞. Il fixe la **sévérité** de l'opérateur booléen (1 représente la sévérité minimale et ∞ est la sévérité maximale).

Les **requêtes** sont des formules composées des **termes pondérés** (valeur réelle dans [0, 1]) et des **opérateurs normés**.

Chapitre II
Modélisation

a. Le modèle booléen (Etendu)

Opérateur normé	n	Sémantique
A n-AND B	∞	min(A, B)
A n-AND B	∈]1, ∞[	si min(A, B)≤1/n alors 0 sinon min(A, B)
A n-AND/n-OR B	1	0
A n-OR B	∈]1, ∞[	si max(A, B)≤1/n alors 0 sinon max(A, B)
A n-OR B	∞	max(A, B)

Chapitre II
Modélisation

a. Le modèle booléen (Etendu)

Exp. 3

Soit la représentation de la pertinence des termes dans 4 documents:

	t_1	t_2	t_3	…	t_n
D1	0,90	0,80	0,09		0,01
D2	0,70	0,40	0,50		0
D3	0,60	0,90	0,80		0,90
D4	0	0,01	0,80		0,90

Chapitre II
Modélisation

a. Le modèle booléen (Etendu)

Exp. 3 (suite)

On définit les opérateurs 2-AND et 2-OR :

a 2-AND b = si min(a,b)≤1/2 alors 0 sinon min(a, b)

a 2-OR b = si max(a, b)≤1/2 alors 0 sinon max(a, b)

On définit aussi les opérateurs ∞-AND et ∞-OR :

a ∞-AND b= min(a, b)

a ∞-OR b = max(a, b)

et soit **R** la requête suivante :

$R = t_1$ ∞-AND (t_2 ∞-OR t_3) 2-AND NOT t_n

Chapitre II
Modélisation

a. Le modèle booléen (Etendu)

Exp. 3 (suite)

Le tableau de vérité de la requête R est:

	t_1	t_2	t_3	…	t_n	NOT t_n	t2 ∞-OR t3	t1 ∞-AND (t2 ∞-OR t3)	R
D1	0.90	0.80	0.09		0.01	0.99	0.80	0.80	0.80
D2	0.70	0.40	0.50		0	1	0.50	0.50	0
D3	0.60	0.90	0.80		0.90	0.10	0.90	0.60	0
D4	0	0.01	0.80		0.90	0.10	0.80	0	0

a. Le modèle booléen (Etendu)

Exp. 3 (suite)

Le calcul fait apparaître que les documents **D_2, D_3 et D_4** ne satisfont pas la requête et **D_1** est pertinent à 80%.

b. Le modèle vectoriel

Dans le modèle vectoriel, le poids $w_{i,j}$ associé au couple (t_i, d_j) est positif et non binaire. Soit le poids des termes de la question $w_{i,q} \geq 0$ associé à la paire (t_i, q).

Le vecteur représentant un document d_j est défini comme suit:

$$\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots w_{k,j})$$

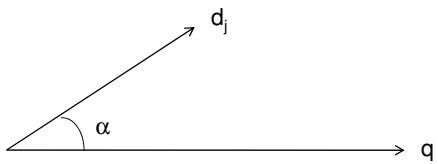
et le vecteur représentant la question est défini comme suit:

$$\vec{q} = (w_{1,q}, w_{2,q}, \ldots w_{k,q})$$

tel que: k est le nombre de termes du corpus

b. Le modèle vectoriel (suite)

Ainsi, un document d_j et une requête sont représentés dans un espace à k dimensions comme suit:



b. Le modèle vectoriel (suite)

La manière la plus simple pour calculer le degré de similarité entre ces deux vecteurs est le cosinus:

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{\left|\vec{d_j}\right| \times \left|\vec{q}\right|} = \frac{\sum_{i=1}^{k} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{k} w_{i,j}^{\ 2}} \times \sqrt{\sum_{i=1}^{k} w_{i,q}^{\ 2}}}$$

Chapitre II
Modélisation

b. Le modèle vectoriel (suite)

Les poids des termes dans les documents d'un corpus peuvent être organisés en **matrice** appelée **termes-documents** dont les **lignes** sont les **termes** du vocabulaire et les **colonnes** sont les **documents** du corpus.

Exp.

	D1	D2	D3		Q
…					
commerce	2.1	0	0		0
…					
enveloppe	0	1.3	2.5		0
…					
possible	0	1.6	0		0
…					
travailler	0	0	1.2		0.9
…					

Chapitre II
Modélisation

b. Le modèle vectoriel (suite)

Exp. suite

Lors de la phase de **recherche**, le vecteur question est **comparé** à chaque colonne de la matrice. On peut utiliser différentes méthodes pour comparer 2 vecteurs, comme le cosinus.

Les colonnes dont le cosinus tend vers 0 désignent les documents satisfaisant la requête. Dans l'exemple précédent, seul le document **D3** est sélectionné.

Chapitre II
Modélisation

Quelques mesures de similarités

Produit scalaire $RSV(\vec{q},\vec{d}) = \vec{q} \bullet \vec{d} = \sum_{i=1}^{n} w_{i,q} \times w_{i,d}$

Cosinus $RSV(\vec{q},\vec{d}) = \frac{\vec{q} \bullet \vec{d}}{|\vec{q}| \bullet |\vec{d}|} = \frac{\sum_{i=1}^{n} w_{i,q} \times w_{i,d}}{\sqrt{\sum_{i=1}^{n} w_{i,q}^{\ 2}} \times \sqrt{\sum_{i=1}^{n} w_{i,d}^{\ 2}}}$

Distance euclidienne $RSV(\vec{q},\vec{d}) = \sqrt{\sum_{i=1}^{n} (w_{i,q} - w_{i,d})^2}$

Chapitre II
Modélisation

Quelques mesures de similarités (suite)

Dice $RSV(\vec{q},\vec{d}) = \frac{2 \cdot \sum_{i=1}^{n} w_{i,q} \times w_{i,d}}{\sum_{i=1}^{n} w_{i,q} + \sum_{i=1}^{n} w_{i,d}}$

Jaccard $RSV(\vec{q},\vec{d}) = \frac{\sum_{i=1}^{n} w_{i,q} \times w_{i,d}}{\sum_{i=1}^{n} w_{i,q} + \sum_{i=1}^{n} w_{i,d} - \sum_{i=1}^{n} w_{i,q} \times w_{i,d}}$

Overlap $RSV(\vec{q},\vec{d}) = \frac{\sum_{i=1}^{n} w_{i,q} \times w_{i,d}}{\min\left(\sum_{i=1}^{n} w_{i,q}, \sum_{i=1}^{n} w_{i,d}\right)}$

Chapitre II
Modélisation

- **Le modèle LSI (Latent Semantic Indexing)**

C'est une extension du modèle vectoriel fondé sur la décomposition en valeurs singulières de la matrice termes-documents.

Dans ce modèle, on utilise le produit **scalaire** pour mesurer la **similarité** entre les documents et la question.

La **ressemblance *R*** entre les documents de la matrice termes-documents ***X*** et le document de la requête ***Q*** est calculée ainsi:

$$R = X^T \bullet Q$$

Problème: Le calcul du vecteur ***R*** est très long car la matrice ***X*** et le vecteur ***Q*** sont des matrices creuses de **grandes tailles**.

Chapitre II
Modélisation

- **Le modèle LSI (suite)**

Pour réduire le temps de calcul de ***R***, on **réduit** la taille de ***X*** et de ***Q*** par une décomposition en composantes principales (Singular Value Decomposition ou SVD).

$$X_{td} = U_{tm} \bullet S_{mm} \bullet (V_{dm})^T$$

tel que:

t: nombre de termes du vocabulaire

d: nombre de documents dans le corpus

m: Le rang de la matrice X ($m \leq \min(t,d)$)

Chapitre II
Modélisation

- **Le modèle LSI (suite)**

Les matrices **U** et **V** sont des matrices orthogonales unitaires $\mathbf{U.U^T = V.V^T = I}$

Les colonnes de la matrice **U** sont les **vecteurs propres** de la matrice symétrique $\mathbf{X.X^T}$.

Chapitre II
Modélisation

- **Le modèle LSI (suite)**

Les colonnes de la matrice **V** sont les **vecteurs propres** de la matrice symétrique $\mathbf{X^T.X}$.

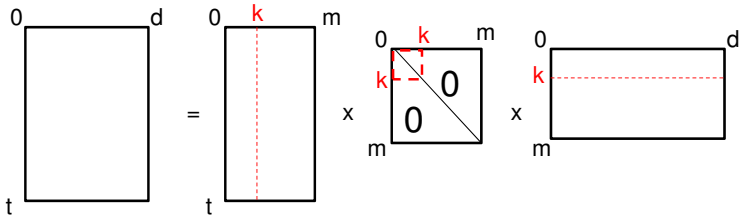
La matrice **S** est la matrice **diagonale triée des valeurs singulières de X (valeurs propres de X)**.

$$S = \begin{bmatrix} \ddots & 0 \\ 0 & \end{bmatrix}$$

Chapitre II
Modélisation

- **Le modèle LSI (suite)**

Cette décomposition permet de représenter les documents dans un espace **réduit** de dimension **k**.



$X_{t*d} = U_{t*m} \times S_{m*m} \times (V_{d*m})^T$

$\approx U_{t*k} \times S_{k*k} \times (V_{d*k})^T$

Chapitre II
Modélisation

- **Le modèle LSI (suite)**

La matrice **U** permet de **faciliter** le calcul du vecteur de pertinence **R** on ne retenant que les *k* premiers vecteurs propres comme suit:

$$R = X^T \bullet Q \Rightarrow R = X^T \bullet (U \bullet U^T) \bullet Q$$

$$R = (X^T \bullet U) \bullet (U^T \bullet Q) \Rightarrow R' = X'^T \bullet Q'$$

au lieu de calculer **R** il suffit de calculer **R'** dans un espace réduit.

$$\begin{cases} (X'_{k*d})^T = (X_{t*d})^T \bullet U_{t*k} \\ Q'_{k*1} = (U_{t*k})^T \bullet Q_{t*1} \end{cases}$$

$$\Rightarrow R'_{d*1} = (X'_{k*d})^T \bullet Q'_{k*1}$$

Chapitre II
Modélisation

- **Le modèle LSI (suite)**

Le vecteur **R'** contient les taux de **ressemblance** entre les documents du corpus et la question. Les valeurs **pertinentes** sont sélectionnées à l'aide d'un **seuil** de filtrage fixé expérimentalement.

Chapitre II
Modélisation

c. Le modèle probabiliste

Ce modèle a était proposé par Steve Roberston en 1974. L'idée est de **trier** les documents d'un corpus en **fonction** de leurs **pertinences** par rapport au document **question**.

Notons:

$P(D,R|Q)$ la **probabilité** pour que le document *D* soit **pertinent** pour une question *Q*

$P(D,\overline{R}|Q)$ la **probabilité** qu'il **ne soit pas pertinent** pour la question *Q*.

c. Le modèle probabiliste (suite)

La probabilité peut être calculée de plusieurs façons. En **général**, on utilise les **termes** du document pour calculer la **pertinence** de ce document par rapport à la question. Pour chaque terme ***i*** on définit un attribut A_i qui peut représenter la **présence** du terme ***i*** de la **question** dans le **document**.

$$P(D,R|Q)=\prod_i P(A_i=a_i,R|Q)$$

$$P(D,\overline{R}|Q)=\prod_i P(A_i=a_i,\overline{R}|Q)$$

où a_i est la valeur de l'attribut A_i .

Pour simplifier, considérons que l'attribut A_i représente la présence (a_i = 1) ou l'absence (a_i = 0) du terme ***i*** de la question dans le document.

c. Le modèle probabiliste (suite)

Les définitions des probabilités deviennent :

$P(A_i=1,R|Q)=P(t_i \text{ présent},R|Q)$: La probabilité pour que le terme t_i de **Q** soit présent dans le document **D** et que ce terme t_i est pertinent pour **D**.

$P(A_i=0,R|Q)=P(t_i \text{ absent},R|Q)$: La probabilité pour que le terme t_i de **Q** soit absent dans le document **D** et que ce terme t_i est pertinent pour **D**.

$P(A_i=1,\overline{R}|Q)=P(t_i \text{ présent},\overline{R}|Q)$: La probabilité pour que le terme t_i de **Q** soit présent dans le document **D** et que ce terme t_i est non pertinent pour **D**.

$P(A_i=0,\overline{R}|Q)=P(t_i \text{ absent},\overline{R}|Q)$: La probabilité pour que le terme t_i de **Q** soit absent dans le document **D** et que ce terme t_i est non pertinent pour **D**.

c. Le modèle probabiliste (suite)

Notons: $P(i\ présent,R|Q)=p_i$ avec p_i est la **probabilité** $P(i\ présent,\overline{R}|Q)=\overline{p_i}$

pour que le terme t_i de la question soit **pertinent et présent** dans le document, et $\overline{p_i}$ est la **probabilité** pour que le terme t_i de la question soit **présent** dans le document mais **pas pertinent**.

c. Le modèle probabiliste (suite)

Pour chaque terme de la question nous calculons un **poids** w_i comme suit :

$$w_i=\log\frac{P(i\ présent,R|Q)\cdot P(i\ absent,\overline{R}|Q)}{P(i\ présent,\overline{R}|Q)\cdot P(i\ absent,R|Q)}$$

avec w_i **augmente** si le terme ***i*** de la question est:
- **soit présent** dans le document et **pertinent**
- **soit absent** du document et **non pertinent**.

w_i **baisse** si le terme ***i*** de la question est:
- **soit présent** dans le document et **non pertinent**,
- **soit absent** du document et **pertinent**.

c. Le modèle probabiliste (suite)

$$\begin{cases} P(i\ absent, R|Q) = 1 - P(i\ présent, R|Q) \\ P(i\ absent, \overline{R}|Q) = 1 - P(i\ présent, \overline{R}|Q) \end{cases} \Leftrightarrow w_i = \log \frac{p_i \cdot (1 - \overline{p_i})}{\overline{p_i} \cdot (1 - p_i)}$$

Les documents du corpus sont **triés** suivant les **valeurs croissantes** de l'expression suivante:

$$W_{D,Q} = \sum_{i=1}^{taille(Q)} \log \frac{p_i (1 - \overline{p_i})}{\overline{p_i} (1 - p_i)}$$

Un **seuil** doit être **fixé** pour pourvoir **décider** quels sont les documents **acceptés**.

c. Le modèle probabiliste (suite)

Le modèle **probabiliste** fournit un résultat **trié** par **ordre** de **pertinence**.

Le modèle probabiliste ne tient **pas compte** de la **position** des termes dans les documents.

La présentation fournie par ce modèle est une **approximation** du contenu d'un document (par exemple, avec une probabilité de 80% on n'est pas sûr que le terme soit pertinent) ce qui **influe** sur la **précision** des résultats.

Le **calcul** de probabilité pour un **grand nombre** de termes peut être **long**.

d. Le modèle basé sur les réseaux de neurones

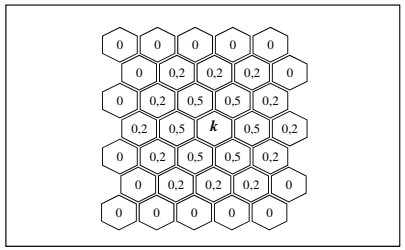
L'idée principale est de **partitionner** le corpus pour **faciliter** la recherche.

Parmi les méthodes de ce modèle on peut citer la plus utilisée avec succès utilisant un **réseau de neurones** à **deux couches**. (Une couche d'entrée et une autre de sortie)

La **couche d'entrée** représente le **vocabulaire** (les termes) $\overrightarrow{w_e} = (x_1, ..., x_n)$ et la **couche de sortie** représente les **partitions voulues**.

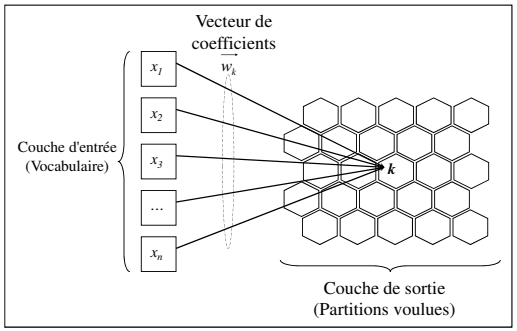
d. Le modèle basé sur les réseaux de neurones (suite)

Les **partitions** sont organisées de sorte qu'une partition soit **entourée** par celles qui lui sont **similaires**.



Les neurones de la couches d'entrée sont **connectés** aux neurones de la couche de sortie par des **connexions** avec des **poids** formant un vecteur w_k

b. Les modèles basés sur les réseaux de neurones (suite)



b. Les modèles basés sur les réseaux de neurones (suite)

Un réseau de neurones est utilisé en **deux phases**:

Phase d'apprentissage: pour faire apprendre au réseau de neurones ce qu'il va faire. Dans notre cas, partitionner le corpus en classes.

Phase de recherche: la recherche d'un document dans le corpus revient à chercher ce document dans un seul neurone (la partition la plus proche à la question)

b. Les modèles basés sur les réseaux de neurones (suite)

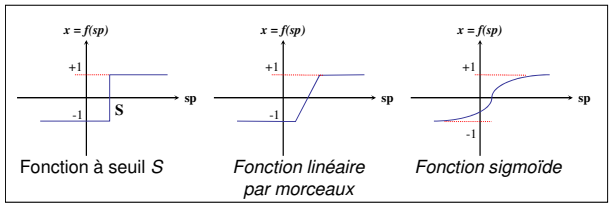
Chaque neurone 'k' possède sont propre vecteur de coefficient $\overrightarrow{w_k}$ et une somme pondérée '**sp_k**'.

$$sp_k = \overrightarrow{w_k} \bullet \overrightarrow{w_e} = \sum_i (w_{ki} \cdot w_{ei})$$

La **définition** du neurone de sortie qui doit **contenir** le document en entrée se fait par le biais d'une **fonction d'activation** utilisant la **somme pondérée**.

b. Les modèles basés sur les réseaux de neurones (suite)

Plusieurs définitions de cette fonction ont été proposées dont les plus utilisées sont:



Fonction d'activation d'un neurone

b. Les modèles basés sur les réseaux de neurones (suite)

On dit qu'un **neurone** est **actif** (ou gagnant) quand il a la valeur d'état $f(sp_k)$ la plus grande pour un vecteur d'entrée

Le **neurone actif** est la **classe candidate** qui doit contenir le **document en entrée**.

Algorithme d'apprentissage (Algorithme de Kohonen):

1. **Initialisation** des coefficients à des valeurs **aléatoires**, ainsi que le taux d'apprentissage **σ**.
2. Présentation d'une entrée $\overrightarrow{w_e}$
3. Trouver le **neurone gagnant** (winner), ou le neurone **actif**.
4. Modifier les poids w_i du neurone gagnant en utilisant le taux d'apprentissage **σ**.
5. Tant que les performances sont **insuffisantes** : Retour à l'étape 3.
6. S'il reste encore des documents dans la base d'apprentissage, aller à l'étape 2.

Algorithme d'apprentissage (Algorithme de Kohonen):

Soit ***E*** la base de documents d'apprentissage;

Soit ***K*** l'ensemble des neurones souhaités dans la couche de sortie;

Pour tout ***e ∈ E*** {

FAIRE {

$$sp_c = \overrightarrow{w_c} \bullet \overrightarrow{w_e} = \sum_i (w_{ci} \cdot w_{ei})$$

Pour tout ***c ∈ K*** {

}

//Recherche du neurone gagnant

$\overrightarrow{w_k} = \overrightarrow{w_k} + \sigma \cdot (\overrightarrow{w_e} - \overrightarrow{w_k})$

Trouver k tel que $\overrightarrow{sp_k}$ est maximal;

$(\overrightarrow{w_k} = \overrightarrow{w_k})$

Calculer $\left|\overrightarrow{w_k} - \overrightarrow{w_e}\right| \geq \varepsilon$;

; // Mettre à jour par

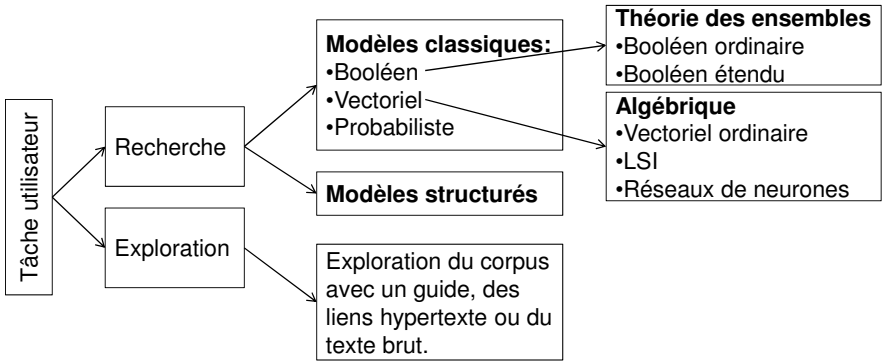
} TANT QUE ( )

}

b. Les modèles basés sur les réseaux de neurones (suite)

Dans la phase de **recherche** le document **question passera** par les même étapes de la phase d'apprentissage pour définir la **classe candidate** (neurone actif) qui doit contenir les **documents** du corpus les plus **proches** à la question.

e. Conclusion



I.3. Bibliographie

- Ricardo Baeza-Yates and Berthier Ribeiro-Neto, "*Modern Information Retrieval*", Addison-Wesley Educational Publishers Inc, 1999.
- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *"An Introduction to Information Retrieval".* Cambridge University Press, 2008.

1. Introduction
2. Problématique
3. Clients, serveurs, courtiers et utilisateurs
4. Méthodes de sélection de serveurs
   a. CORI
   b. vGlOSS & bGlOSS
   c. CVV

1. Introduction

Les systèmes de recherche d'information ont évolués pour toucher au monde **distribué**.

Avec cette évolution, **l'utilisateur** (le client) peut **lancer** une **recherche** depuis un endroit **loin** du **serveur** ou se trouve l'index du corpus.

2. Problématique

Le **problème** principal de la recherche d'information distribuée se pose quand l'utilisateur accède à **plusieurs** documents **localisés** dans **différents serveurs**.

L'utilisateur **ne connait** pas ces serveurs qui contiennent les documents pertinents. Donc, il doit **lancer** la même **requête** sur **tous les serveurs** qui sont sur le réseau.

Les **nouveaux** serveurs **ne sont pas connus** par l'utilisateur.

3. Clients, serveurs, courtiers et utilisateurs

Pour résoudre le problème des systèmes de recherche d'information distribuée, l'idée est d'ajouter un **client de recherche** sophistiqué appelé « **courtier** ».

Le **courtier** reçoit la **requête** de l'utilisateur, collecte la liste des **serveurs** accessibles et génère une liste de **serveurs pertinents**.

La recherche de la **meilleur sélection** de serveurs est un **problème d'optimisation complexe**.

3. Clients, serveurs, courtiers et utilisateurs (suite)

Pendant la recherche, le courtier **applique** la requête **q** de l'utilisateur à l'ensemble **S** des **serveurs** de la meilleur sélection et obtient des **listes de résultats R_1, … $R_{|S|}$**.

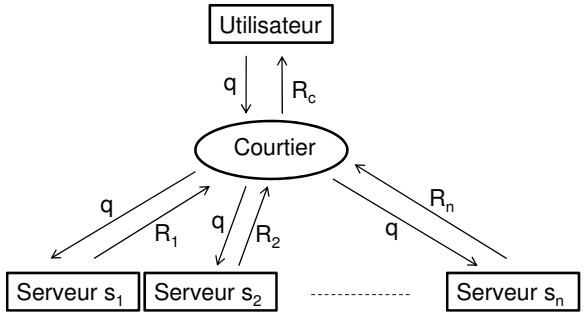
Chaque **liste** de résultats $R_i = <L_i, o_i>$ contenant un ensemble de documents L_i et un ordre o_i.

Au cours de la **fusion** des résultats, le courtier **combine** les résultats R_1, …, $R_{|S|}$ pour générer une **seule liste** de résultats combinés: $R_c = <L_c, o_c>$ tel que:

$L_c = L_1 \cup \ldots \cup L_{|S|}$

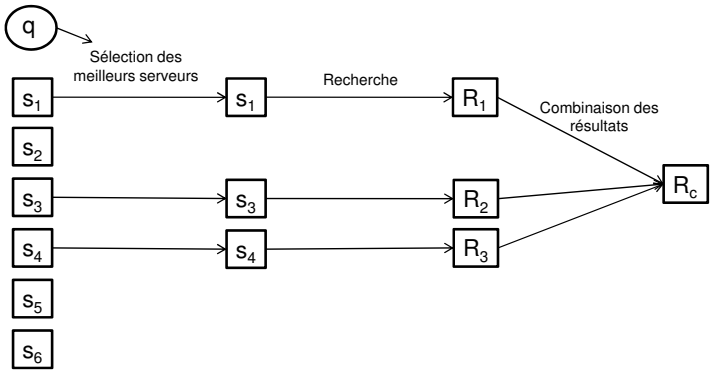
et o_c est le classement.

3. Clients, serveurs, courtiers et utilisateurs (suite)



Scénario de recherche

3. Clients, serveurs, courtiers et utilisateurs (suite)



Processus de recherche d'un courtier

4. Méthodes de sélection de serveurs

De nombreuses méthodes de sélection de serveurs sont des méthodes de **classement** selon la requête utilisateur **q**.

Les méthodes les plus connues sont: CORI, bGlOSS, vGlOSS, CVV.

4. Méthodes de sélection de serveurs

a. CORI

C'est un **algorithme** de **sélection** de **collections**. Il est basé sur les réseaux d'inférence bayesiens.

Dans CORI, les **similarités** entre la requête utilisateur **q** et un ensemble de collections de documents sont calculées dans le but de trier ces collections.

La **requête** est ensuite **envoyée** à tout les serveurs sélectionnés pour rechercher les documents **pertinents** de chaque **serveur**. Ces documents pertinents seront ensuite **combinés**.

a. CORI (suite)

La similarité entre la requête **q** et les documents du serveur $\mathbf{s_i}$ est calculée par l'équation suivante:

$$CORI(q, s_i) = \frac{\sum_{k=1}^{Q} d_b + (1 - d_b) \times T_{i,k} \times I_k}{Q}$$

tel que:

$\mathbf{d_b}$ est la **croyance** pas défaut (default belief) initialisée à **0.4**

Q est le **nombre** de termes de la **question** sans redondance

a. CORI (suite)

$$T_{i,k} = d_t + (1-d_t) \times \frac{\log(DF_{i,k} + 0.5)}{\log(DF_i^{\max} + 1.0)}$$

$$I_k = \frac{\log\left(\frac{|S|+0.5}{SF_k}\right)}{\log(|S|+1.0)}$$

tel que:

$T_{i,k}$ Poids du terme $\mathbf{t_k}$ dans le corpus du serveur $\mathbf{s_i}$.

I_k est la fréquence inverse de la collection (the inverse collection frequency).

a. CORI (suite)

d_t est la fréquence pas défaut du terme (default term frequency) initialisée aussi à **0.4**

$DF_{i,k}$ est le nombre de documents du serveur $\mathbf{s_i}$ contenant le terme $\mathbf{t_k}$.

DF_i^{max} est le DF maximal de tous les termes dans le serveur $\mathbf{s_i}$. = max($Df_{i,k}$)

SF_k est le nombre de serveurs qui ont $\mathbf{Df_{i,k} > 0}$

S est l'ensemble de serveurs sur le réseau (non pas l'ensemble des serveurs pertinents).

a. CORI (suite)

Exp.

Soit un système distribué contenant 4 serveurs:

s1: D1(a,b,c), D2(d,e,f), D3(g,h,i)

s2: D4(j,k,l), D5(m,n,o), D6(p,q,r)

s3: D7(s,t,u), D8(v,w,x), D9(y,z,a)

s4: D10(b,c,d), D11(e,f,g), D12(h,i,j)

et soit la question q(a, m, z, j)

En utilisant la méthode **CORI**, trouvez les serveurs pertinents pour la question **q** en utilisant le seuil **2,0**.

Exp. (suite)

$$CORI(q, s_1) = \frac{\sum_{k=1}^{4} 0.4 + (1-0.4) \times T_{1,k} \times I_k}{4}$$

$$T_{1,a} = 0.4 + (1-0.4) \times \frac{\log(1+0.5)}{\log(1+1.0)} = 0.75$$

$$I_a = \frac{\log\left(\frac{4+0.5}{2}\right)}{\log(4+1.0)} = 0.504$$

Exp. (suite)

T2j	0,7509775		DF2j	1			
T3a	0,7509775		DF3a	1	DF3max	1	
T3m	-0,2		DF3m	0			
T3z	0,7509775		DF3z	1			
T3j	-0,2		DF3j	0			
T4a	-0,2		DF4a	0	DF4max	1	
T4m	-0,2		DF4m	0			
T4z	-0,2		DF4z	0			
T4j	0,7509775		DF4j	1			

Ia	0,50385927	**CORI(q,s1)**	**1,288**
Im	0,93453583	**CORI(q,s2)**	**1,605**
Iz	0,93453583	**CORI(q,s3)**	**1,821**
Ij	0,50385927	**CORI(q,s4)**	**1,072**

Exp. (suite)

Donc les serveurs pertinents sont: $\mathbf{s_2}$ et $\mathbf{s_3}$ puisqu'ils possèdent une grande similarité par rapport à la question **q**.

b. vGlOSS et bGlOSS (vector-space/boolean Glossary Of Servers Server)

GlOSS est un **serveur** contenant un **index** de serveurs pertinents à une question **q**.

vGlOSS est basée sur le modèle vectoriel. Dans cette méthode on commence par définir la mesure **Goodness** qui détermine quels sont les meilleurs serveurs pour la question **q**.

$$Goodness(l,q,s) = \sum_{d \in Rang(l,q,s)} sim(q,d)$$

tel que: ***l*** est le seuil de similarité.

sim(q,d): la similarité entre **q** et le document **d**.

Rang(l,q,s) = $\{d \in s \mid sim(d,q) > l\}$ le **rang idéal** des serveurs.

b. vGlOSS et bGlOSS (suite)

On détermine après **l'ensemble Ideal(l)** de serveurs pertinents en triant les serveurs selon leurs **Goodness** par rapport à la question **q** et on prenant ceux qui **dépassent l**.

Exp. Soit deux serveurs s_1 et s_2 et la question q. et soit les réponses reçus par ces serveurs on posant comme question q:

s1: (d_1^1, 0.9), (d_2^1, 0.9), (d_3^1, 0.1)

s2: (d_1^2, 0.8), (d_2^2, 0.4), (d_3^2, 0.3) , (d_4^2, 0.1)

On prenant le seuil **l=0.2** nous aurons:

$$Goodness(0.2,q,s_1) = 0.9 + 0.9 = 1.8$$

$$Goodness(0.2,q,s_2) = 0.8 + 0.4 + 0.3 = 1.5$$

b. vGlOSS et bGlOSS (suite)

parce que $\mathbf{s_1}$ possède **2** documents avec une similarité>0.2 et $\mathbf{s_2}$ possède **3** documents dont la similarité > 0.2

et: Ideal(1.2) = $\{s_1, s_2\}$

La mesure Goodness n'est pas optimisée. Pour cela, vGlOSS cherche à récupérer plus d'information sur les contenus des serveurs.

L'une des solutions proposées est d'utiliser les matrices:

F = (f_{ij}) : $\mathbf{f_{ij}}$ est le nombre de documents dans $\mathbf{s_i}$ contenant $\mathbf{t_j}$.

W = (w_{ij}) : $\mathbf{w_{ij}}$ est la somme des poids du terme $\mathbf{t_j}$ dans tous les documents du serveur $\mathbf{s_i}$.

b. vGlOSS et bGlOSS (suite)

Pour obtenir $\mathbf{f_{i*}}$ et $\mathbf{w_{i*}}$ pour le serveur $\mathbf{s_i}$, un programme appelé « **collector** » est lancé **périodiquement** pour récupérer ces informations et les envoyer au serveur vGlOSS.

Exp. Soit un serveur **s** et le terme « ordinateur ». Et soit les documents suivants du serveur **s** contenant le terme « ordinateur » avec leurs poids :

ordinateur : $(d_1, 0.8)$, $(d_2, 0.7)$, $(d_3, 0.9)$, $(d_8, 0.9)$

Limite de vGlOSS:

Le **collector** n'envoie pas toutes ces informations au serveur vGlOSS, mais il l'informe juste qu'il y a **4** documents contenant le terme « ordinateur » et que la somme des poids est **3.3**

b. vGlOSS et bGlOSS (suite)

Pour faire face à cette limite, on suppose que:

Le poids d'un terme est distribué uniformément sur tous les documents qui le contiennent. Ce qui veut dire qu'un terme t_j aura le poids (w_{ij}/f_{ij}) dans tous les documents du serveur s_i qui contiennent le terme t_j.

Avec cette supposition, deux scénarios sont proposés: scénario avec **grande-corrélation** et scénario **disjoint**.

Scénario avec grande-corrélation

Ce scénario propose une supposition:

Si t_1 et t_2 des termes de q et $0 < f_{i1} \leq f_{i2}$, alors les documents de s_i contenant t_1 ils contiennent aussi t_2.

b. vGlOSS et bGlOSS (suite)

Exp1. Soit le serveur $\mathbf{s_i}$ et la question **q** = "computer science department", et soit $f_{i1}=2$, $f_{i2}=9$, et $f_{i3}=10$.

vGlOSS avec le scénario grande-corrélation suppose que les 2 documents contenant le terme "computer" contiennent aussi les termes "science" et "department".

Aussi les 7 documents contenant le terme "science" et sans "computer" contiennent aussi "department". Et finalement, 1 seul document contient le terme "department" sans "computer" et "science".

b. vGlOSS et bGlOSS (suite)

Exp2. (suite exp 1.) soit les poids des termes de la question **q** dans le serveur $\mathbf{s_i}$ sont: w_{i1}=0.45, w_{i2}=0.2, w_{i3}=0.9

Selon la supposition de vGlOSS, le poids du terme "computer" dans les deux documents contenant ce terme est: 0.45/2 = 0.225

Le poids du terme "science" dans les 9 documents contenant ce terme est: 0.2/9 = 0.022

Le poids du terme "department" dans les 10 documents contenant ce terme est: 0.9/10 = 0.09

b. vGlOSS et bGlOSS (suite)

Après on calcul l'estimation de similarité entre la question et chaque serveur selon un certain seuil:

$$Estimate(l,q,s_i) = \sum_{j=1,..,p} \left(f_{ij} - f_{i(j-1)}\right) \times sim_j$$

tel que: $sim_j = \sum_{k=j,..,n} q_k \times \frac{w_{ik}}{f_{ik}}$

$\mathbf{q_k}$ est la fréquence de $\mathbf{t_k}$ dans **q.**

p est l'ordre du terme $\mathbf{t_p}$ dans la question défini comme suit: $\mathbf{sim_p > l}$ **et** $\mathbf{sim_{p+1} \le l}$ (les f_{ij} doivent être triés $f_{i0}<f_{i1}<\ldots$)

n: Nombre de termes dans la question **q**.

b. vGlOSS et bGlOSS (suite)

Exp 3. (suite exp 2.) soit les poids des trois termes de la question **q** sont = 1.

Selon la 2ème supposition, les 2 documents avec le terme "computer" contiennent aussi les termes "science" et "department". Et selon la 1ère supposition, la similarité de ces 2 documents par rapport à q est: $\mathbf{sim_1}$ = 0.45/2 + 0.2/9 + 0.9/10 = 0.337 > l=0.2

Si le seuil l=0.2 alors ces 2 documents sont acceptés.

La similarité des 7 documents contenant "science" et "department" et sans "computer" par rapport à **q** est: $\mathbf{sim_2}$ = 0.2/9 + 0.9/10 = 0.112 < l=0.2

Ces 7 documents sont ignorés.

À partir de là, (sim1 > l et sim2 < l) nous aurons **p = 1**.

b. vGlOSS et bGlOSS (suite)

Exp 3. (suite exp 2.) donc :

$$Estimate(0.2,q,s_i) = \left(f_{i1} - f_{i0}\right) \times sim_1$$

$$= f_{i1} \times sim_1 = 2 \times 0.337 = 0{,}674$$

Les serveurs sélectionnés comme serveurs pertinents sont ceux qui ont une bonne valeur ***Estimate***.

b. vGlOSS et bGlOSS (suite)

Scénario disjoint

Ce scénario suppose que:

Si deux termes t_1, t_2 apparient ensemble dans une question q, ces termes ne doivent pas figurer ensemble dans un document du serveur sélectionné.

L'estimation de la similarité entre un serveur et la question est:

$$Estimate(l, q, s_i) = \sum_{k=1,..,n \,|\, (f_{ik}>0) \wedge \left(q_k \times \frac{w_{ik}}{f_{ik}} > l\right)} q_k \times w_{ik}$$

q_k, n, w_{ik}, f_{ik}: ont la même définition comme dans l'autre scénario.

b. vGlOSS et bGlOSS (suite)

Exp 4. (suite exp 3.)

Selon la supposition du scénario disjoint, il y a 2 documents contenant le terme "computer" et sans les autres termes, 9 documents contenant le terme "science" et sans les autres termes et 10 documents contenant le terme "department" et sans les autres termes.

Les documents du 1er groupe ont la similarité = 0.45/2 = 0.225, ils sont donc acceptés.

Les documents du 2nd et 3ème groupes sont ignorés parce que leurs similarités sont: 0.2/9=0.022 et 0.9/10=0.09

L'estimation de la similarité entre le serveur et la question est:

b. vGlOSS et bGlOSS (suite)

Exp 4. (suite exp 3.)

$$Estimate(0.2, q, s_i) = w_{i1} = 0.45$$

parce que le seule terme qui satisfait la condition:

$$k = 1,.., n \,|\, (f_{ik}>0) \wedge \left(q_k \times \frac{w_{ik}}{f_{ik}} > l\right)$$

est le terme t_1 de la question « computer ».

Les serveurs sélectionnés comme serveurs pertinents sont ceux qui ont une bonne valeur *Estimate*.

b. vGlOSS et bGlOSS (suite)

bGlOSS est une version de GlOSS supportant le **modèle booléen** de la recherche documentaire.

Puisque la notion de similarité entre document et question n'existe pas dans le modèle booléen (le document satisfait ou pas la question), bGlOSS à **besoin** juste de la matrice **$F=(f_{ij})$** tel que **f_{ij}** est le nombre de documents du serveur **s_i** contenant le terme **t_j**.

b. vGlOSS et bGlOSS (suite)

Exp. Soit 3 serveurs s_1, s_2, s_3 et supposant que bGlOSS a collecté les statistiques de la table suivante:

	s_1	s_2	s_3
Nb documents	100	1000	200
Nb documents contenant le terme 't_1'	40	500	10
Nb documents contenant le terme 't_2'	5	40	0

Et soit la requête: $q = t_1 \wedge t_2$

b. vGlOSS et bGlOSS (suite)

Exp. (suite) Il est claire que le serveur $\mathbf{s_3}$ ne contient pas de documents satisfont la requête **q**.

Pour les deux autres serveurs, bGlOSS doit faire des **suppositions** pour calculer le nombre de documents pertinents. Il existe différents **estimateurs** qui peuvent être utilisés pour faire ces **suppositions**.

Un de ces **estimateurs** qu'on va étudier dans cette partie s'appel **Ind** (indépendance) qui **suppose** que les termes **apparient** dans les documents **indépendamment**.

b. vGlOSS et bGlOSS (suite)

Exp. (suite)

L'estimateur **Ind** estime le résultat comme suit:

$\mathbf{s_1}$ contient 100 documents, 40 de ces documents contiennent le terme $\mathbf{t_1}$. Donc la probabilité pour qu'un document de $\mathbf{s_1}$ contient $\mathbf{t_1}$ est 40/100.

De même, la probabilité pour qu'un document de $\mathbf{s_1}$ contient $\mathbf{t_2}$ est 5/100.

b. vGlOSS et bGlOSS (suite)

Exp. (suite) Donc on peut estimer le nombre de documents de $\mathbf{s_1}$, $\mathbf{s_2}$, $\mathbf{s_3}$ qui satisfont la requête **q** selon l'équation suivante:

$$Estimate_{Ind}(t_1 \wedge \ldots \wedge t_n, s_i) = \frac{\prod_{j=1}^{n} f_{ij}}{|s_i|^{n-1}}$$

b. vGlOSS et bGlOSS (suite)

Exp. (suite)

$$Estimate_{ind}(q, s_1) = \frac{40 \times 5}{100} = 2$$

$$Estimate_{ind}(q, s_2) = \frac{500 \times 40}{1000} = 20$$

$$Estimate_{ind}(q, s_3) = \frac{10 \times 0}{200} = 0$$

Donc le meilleur serveur par indépendance (Ind) est $\mathbf{s_2}$, suivi par $\mathbf{s_1}$.

c. CVV (Cue validity Variance)

La méthode de sélection de serveurs CVV (Variance de validité de repérage) utilise une mesure $\mathbf{CVV_j}$ pour chaque terme $\mathbf{t_j}$ de la question comme suit:

$$CVV_j = \frac{\sum_{i=1}^{|S|} \left(CV_{ij} - \overline{CV_j}\right)^2}{|S|}$$

Tel que: **S** est l'ensemble des serveurs.

$\mathbf{CV_{ij}}$ est la variance de repérage du terme $\mathbf{t_j}$ dans le serveur $\mathbf{s_i}$, se calcul comme suit:

c. CVV (suite)

$$CV_{ij} = \frac{\frac{DF_{ij}}{SS_i}}{\frac{DF_{ij}}{SS_i} + \frac{\sum_{k \neq i}^{|S|} DF_{kj}}{\sum_{k \neq i}^{|S|} SS_k}}$$

Tel que: $\mathbf{SS_i}$ est le nombre de documents dans le serveur $\mathbf{s_i}$.

$\mathbf{DF_{ij}}$ est le nombre de documents du serveur $\mathbf{s_i}$ contenant le terme $\mathbf{t_j}$.

c. CVV (suite)

$$\overline{CV_j} = \frac{\sum_{i=1}^{|S|} CV_{ij}}{|S|}$$

c. CVV (suite)

La méthode CVV permet de sélectionner les serveurs pertinents à une question **q** en calculant un score **scr_i** pour chaque serveur comme suit:

$$scr_i = \sum_{j=1}^{Q} CVV_j \times DF_{ij}$$

Tel que: **Q** est le nombre de termes de **q**.

DF_{ij} est le nombre de documents du serveur **s_i** contenant le terme **t_j**.

Exp. Soit 3 serveurs s_1, s_2, s_3 tel que:

	s_1	s_2	s_3
Nb documents	100	1000	200
Nb documents contenant le terme 't_1'	40	500	10
Nb documents contenant le terme 't_2'	5	40	0

Et soit la requête: $q = t_1 \wedge t_2$

Exp: (suite)

DF_{11}=40

CV_{11}=(40/100)/((40/100)+((500+10)/(1000+200)))=0.485

DF_{12}=5	CV_{12}=0.6
DF_{21}=500	CV_{21}=0.74
DF_{22}=40	CV_{22}=0.7
DF_{31}=10	CV_{31}=0.009
DF_{32}=0	CV_{32}=0

Exp: (suite)

$$\overline{CV_1} = \frac{0.485 + 0.74 + 0.009}{3} = 0.41$$

$$\overline{CV_2} = \frac{0.6 + 0.7 + 0}{3} = 0.43$$

$$CVV_1 = \frac{\left(CV_{11} - \overline{CV_1}\right)^2 + \left(CV_{21} - \overline{CV_1}\right)^2 + \left(CV_{31} - \overline{CV_1}\right)^2}{3} = 0.092$$

$$CVV_2 = \frac{\left(CV_{12} - \overline{CV_2}\right)^2 + \left(CV_{22} - \overline{CV_2}\right)^2 + \left(CV_{32} - \overline{CV_2}\right)^2}{3} = 0.096$$

Exp: (suite)

$$scr_1 = CVV_1 \times DF_{11} + CVV_2 \times DF_{12} = 4.16$$

$$scr_2 = CVV_1 \times DF_{21} + CVV_2 \times DF_{22} = 49.84$$

$$scr_3 = CVV_1 \times DF_{31} + CVV_2 \times DF_{32} = 0.92$$

Selon les scores, on trouve que $\mathbf{s_2}$ est le plus pertinent, puis le $\mathbf{s_1}$.