Recherche d'information

Modèles en Recherche d'Information

Cours Master Recherche Paris 13
Recherche et extraction d'information

A. Rozenknop
source : Romaric Besançon
CEA-LIST/LIC2M

Modèles en RI

Qu'est-ce qu'un modèle en RI ?

- décrire le processus computationnel
  - *e.g.* comment les documents retournés sont ordonnés
  - comment les documents sont stockés relève de l'implémentation (→ index)
- décrire le processus humain
  - *e.g.* le besoin d'information, l'interaction
- Les variables: documents, requêtes, termes d'index, jugements de pertinence, utilisateurs, besoin d'information
- définition implicite ou explicite de la pertinence

Modèles en RI

Modèle booléen

Modèles vectoriels

Modèles probabilistes

Modèles de langue

Autres modèles

réseaux d'inférences, modèles logiques, modèles neuronaux, modèles génétiques...

Exact ou meilleur

appariement exact ou meilleur appariement (*exact match* vs. *best match*)

exact match

critères de pertinence précis

un document correspond ou non à ces critères

le résultat est un ensemble de documents non ordonnés

best match

la requête décrit le meilleur document voulu

le résultat est un ensemble de documents triés par pertinence

Exact ou meilleur

best-match a en général de meilleurs performances (meilleurs documents en premier)

exact-match encore souvent présent dans les systèmes commerciaux (avec des critères de tri supplémentaires)

exact-match

efficace
prévisible, explicable
requêtes structurées
demandes précises

requêtes structurées difficiles à écrire
la difficulté s'accroît avec la taille de la collection
lexique des requêtes=lexique d'indexation
précision correcte → rappel faible
best-match toujours meilleur

Modèle booléen

le modèle *exact match* le plus commun

une requête est une expression logique

termes

opérateurs booléens ET / OU / SAUF

« *recherche* ET *information* ET *modèles* SAUF *indexation* »

Remarque: d'autres modèles (non booléens) utilisent un formalisme booléen pour les requêtes (traité de façon différente)

Modèle booléen

mode d'appariement exact

R : fonction de pertinence (*Relevance*) d'un couple *(document,requête)* :

$$R(D,t)=1 \; si \; t \in D$$
$$R(D,t)=0 \; si \; t \notin D$$

$$R(D,t_1 \; ET \; t_2)=R(D,t_1)\times R(D,t_2)$$
$$R(D,t_1 \; OU \; t_2)=R(D,t_1)+R(D,t_2)-R(D,t_1)\times R(D,t_2)$$
$$R(D,t_1 \; SAUF \; t_2)=R(D,t_1).(1-R(D,t_2))$$

Extensions du modèle booléen

'**?**' => n'importe quel caractère

« *base*? ET *données* »

baser les données, *bases de données*

'*' => troncature

« *ba** ET *données* »

bases de données, *banque de données*

opérateur de proximité (ADJ ou NEAR)

« *recherche ADJ(2) information* »

recherche automatique d'information

Exemple de modèle booléen

système WESTLAW (sur textes de lois)

« *Are there any cases which discuss negligent maintenance or failure to maintain aids to navigation such as lights, buoys, or channel markers* ? »

NEGLECT! FAIL! NEGLIG! /5 MAINT! REPAIR! /P NAVIGAT! /5 AID EQUIP! LIGHT BUOY "CHANNEL MARKER"

!	→ troncature
/*n*	→ proximité
/P	→ même paragraphe

requêtes complexes et longues

pour des spécialistes

Limites du modèle booléen

écrire une requête booléenne est difficile

le modèle ne permet pas de classer les documents retournés par le système

le modèle ne permet pas de retourner un document s'il ne contient qu'une partie des mots de la requête (si le connecteur ET est utilisé)

Modèle booléen et logique floue

Logique floue (Lotfi Zadeh)

Valeurs de vérité entre 0 et 1

Fonction de pondération d'un terme *t* dans le document *D*

$$R(D,t) \in [0,1]$$

$$R(D,t_1 \, ET \, t_2) = \min(R(D,t_1), R(D,t_2))$$
$$R(D,t_1 \, OU \, t_2) = \max(R(D,t_1), R(D,t_2))$$
$$R(D,NOT \, t_1) = 1 - R(D,t_1)$$

Permet de classer les documents

Utilisation de l'index dans un modèle booléen

Algorithme :

$D \leftarrow \{\}$

pour chaque terme t_i de la requête

récupérer l'ensemble D_i des documents d_{ij} tels que $t_i \in d_{ij}$

si opérateur ET

$D \leftarrow D \cap D_i$

si opérateur OU

$D \leftarrow D \cup D_i$

si opérateur SAUF

$D \leftarrow D \setminus D_i$

Modèle vectoriel

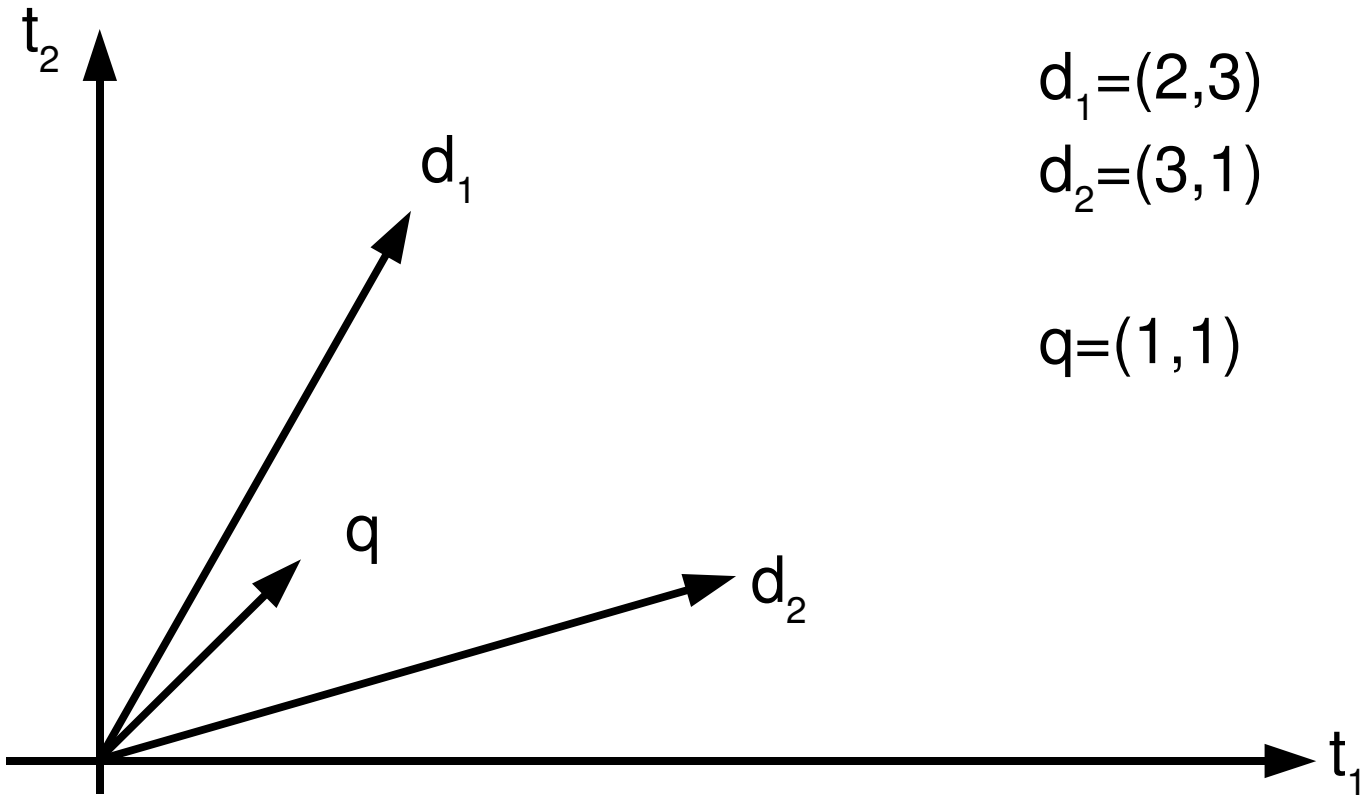
créé au début des années 70 par Gerard Salton : système SMART (*System for the Mechanical Analysis and Retrieval of Text*)

le modèle *best-match* le plus commun

- requête en texte libre
- documents ordonnés dans la réponse

les documents et la requête sont représentés comme des vecteurs dans un espace vectoriel

Modèle vectoriel



$d_1=(2,3)$

$d_2=(3,1)$

$q=(1,1)$

Pertinence

La pertinence est mesurée par une *mesure de similarité* sur l'espace vectoriel de représentation

produit scalaire

$$s(d,q)=\sum_{i=1}^{n} d_i \times q_i$$

avec des valeurs binaires de présence/absence dans les vecteurs ➜ taille de l'intersection entre q et D

Matrice d'occurrence

termes d'index t_1, t_2 ... t_n

documents d_1, d_2 ... d_m

La base documentaire peut se représenter par une matrice D :

$$D = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \end{pmatrix} = \begin{pmatrix} t_1 & t_2 & t_3 & \dots & t_n \\ 1 & 0 & 0 & 1 & \dots & 1 \\ 0 & 1 & 0 & 1 & \dots & 0 \\ 0 & 0 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Pondération des termes

L'importance d'un terme dans un document (*la valeur de la composante du vecteur représentant le document*) dépend de trois choses :

- l'importance du terme dans le document (pondération locale)
- l'importance du terme dans la collection (pondération globale)
- l'importance du document (normalisation en fonction de la taille du document)

Il y a plusieurs possibilités pour calculer chacune de ces pondérations

Pondération locale

facteur *tf* (*term frequency*)

facteur binaire: présence/absence du terme dans le document

$$tf = \begin{cases} 1 \ \ si \ \ t \in D \\ 0 \ \ si \ \ t \notin D \end{cases}$$

facteur fréquentiel: fréquence du terme dans le document (nombre d'occurrences)

$$tf = \text{freq}(t, d)$$

facteur fréquentiel normalisé

$$tf = \frac{\text{freq}(t, d)}{\max_i \text{freq}(t_i, d)}$$

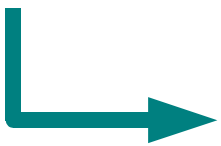
Pondération locale

facteur logarithmique

$$tf = 1 + \log(\text{freq}(t,d)) \ si \ \text{freq}(t,d) \neq 0$$

➪ un document qui contient un grand nombre de fois un terme n'est pas plus pertinent qu'un document qui contient un petit nombre de fois plusieurs termes de la requête

d_1=(2,3)
d_2=(10,0) ➜ s_1=5, s_2=**10**
q =(1,1)



d'_1=(1+log(2),1+log(3))=(1.7,2.1)
d'_2=(1+log(10),0)=(2.3,0)
q =(1,1)
➜ s_1=**3.8**, s_2=2.3

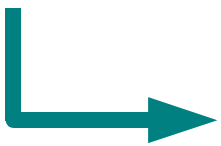
Pondération locale

un autre facteur logarithmique (normalisé)

$$tf = \frac{1+\log(\text{freq}(t,d))}{1+\log(\text{avg}_i \text{freq}(t_i,d))} \; si \; \text{freq}(t,d) \neq 0$$

d_1=(2,3)
d_2=(10,0)
q =(1,1)

➜ s_1=5
s_2=**10**



d'_1=(0.89,1.09)
d'_2=(0.88,0)
q =(1,1)

➜ s_1=1.98
s_2=0.88

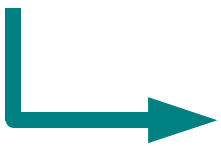
Pondération locale

facteur augmenté

$$tf = 0.5 + 0.5 \times \frac{\text{freq}(t, d)}{max_i \, \text{freq}(t_i, d)}$$

➪ même idée que pour le facteur logarithmique: poids minimal de 0.5, poids maximal de 1

d_1=(2,3)
d_2=(10,0) ➜ s_1=5, s_2=**10**
q =(1,1)



d'_1=(0.83,1)
d'_2=(1,0) ➜ s_1=**1.83**, s_2=1
q =(1,1)

Pondération globale

facteur *idf* (*inverse document frequency*)

importance du terme dans la collection: donner un poids plus important pour les termes moins fréquents (plus discriminants)

$$idf(t) = 1 + \log\left(\frac{N}{df(t)}\right)$$

où N est la taille de la collection
et $df(t)$ est le nombre de documents qui contiennent t
(*fréquence en document*)

pondération de l'espace vectoriel de représentation

Pondération globale

facteur *idf* probabiliste

dérivés des modèles probabilistes (présentés plus loin)

$$idf(t) = 1 + \log\left(\frac{N - df(t)}{df(t)}\right)$$

Longueur des documents

facteur de normalisation en fonction de la longueur des documents

certains documents sont plus longs que d'autres

verbosité : les documents plus longs traitent plus longuement du même thème => termes plus fréquents

normalisation du facteur *tf*

variations thématiques : les documents plus longs traitent plus de thèmes => terminologie plus variée => meilleur score

si plusieurs documents ont la même pertinence, on préfère le plus court (plus concis)

Normalisation

normalisation par la somme des *tf* (norme L1)

$$N(d) = \sum_{i=1}^{n} d_i$$

norme euclidienne (L2)

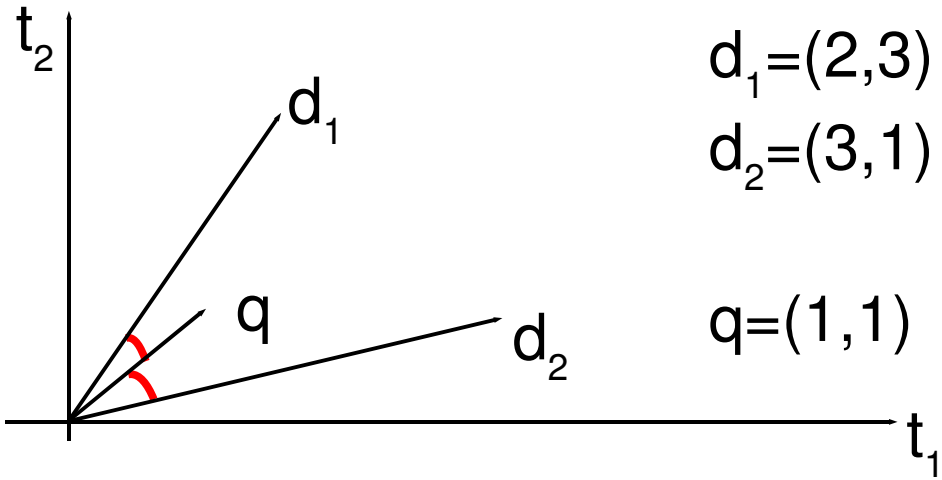
$$N(d) = \sqrt{\sum_{i=1}^{n} d_i^2}$$

Cosinus

avec la norme L2, la formule de similarité est

$$s(d,q)=\frac{\sum_{i=1}^{n} d_i \times q_i}{\sqrt{\sum_{i=1}^{n} d_i^2}\sqrt{\sum_{i=1}^{n} q_i^2}}$$

c'est le *cosinus* de l'angle entre les deux vecteurs du document et de la requête



Normalisation à pivot

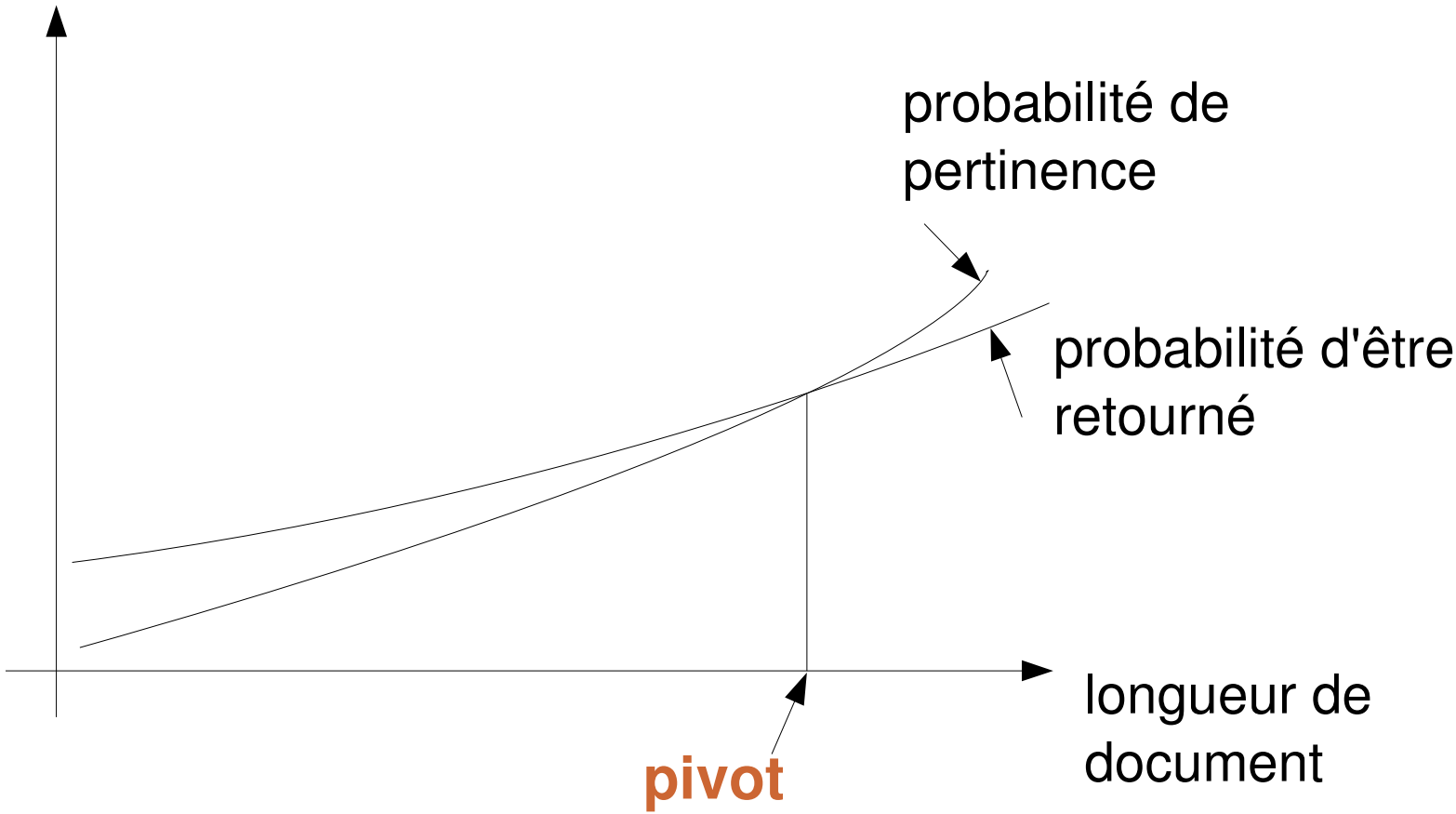
Avec la normalisation du cosinus, les documents les plus courts sont préférés

des schémas de pondération plus avancés

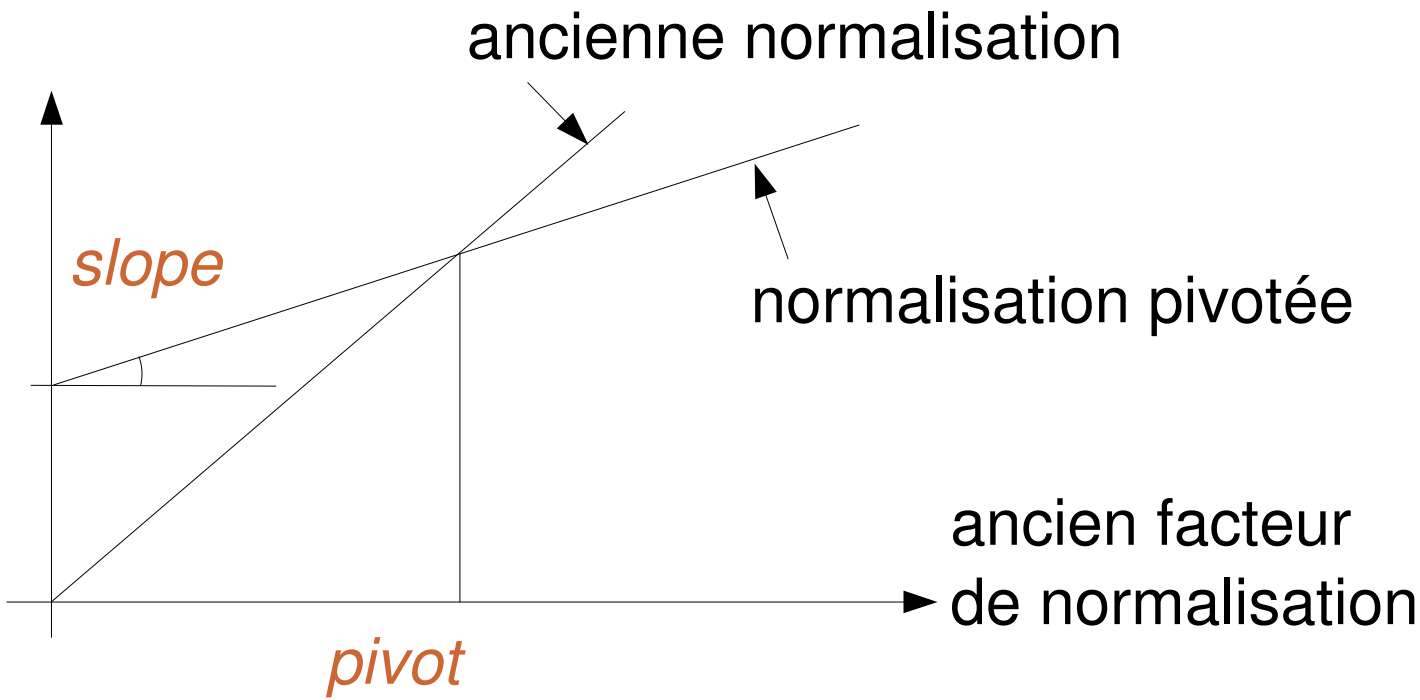
Singhal et al se sont posés la question :

- est-ce que la pertinence et la longueur des documents sont indépendants ?
- retourner les documents d'une certaine longueur avec une probabilité proportionnelle à la probabilité qu'un document de cette longueur soit pertinent

Normalisation à pivot



Normalisation à pivot



$$norm_{new} = 1 + \frac{slope}{(1 - slope) \times pivot} \times norm_{old}$$

Normalisation à pivot

pivot et *slope* sont des paramètres du modèles

ils peuvent être estimés par apprentissage

pivot est parfois fixé à une valeur dépendant de la collection (par exemple, la valeur moyenne de la normalisation précédente sur la collection)
➥ dans ce cas, le paramètre *slope* est estimé de façon *ad hoc* en fonction des évaluations

la normalisation initiale souvent utilisée est le nombre de termes différents du document (noté *NT(d)*)
➜ *normalisation unique à pivot*

$$norm_{new} = (1 - slope) \times pivot + slope \times NT(d)$$

Combinaison des pondérations

Notation de la combinaison des pondérations choisie

ABC.DEF

où

- A = pondération locale dans le document
- B = pondération globale dans le document
- C = normalisation dans le document
- D = pondération locale dans la requête
- E = pondération globale dans la requête
- F = normalisation dans la requête

Combinaison des pondérations

Notations

pondération locale

- b = facteur binaire
- n = facteur fréquentiel
- m = facteur fréquentiel normalisé
- l = facteur logarithmique
- L = facteur logarithmique normalisé
- a = facteur augmenté

pondération globale

- n = pas de pondération (=1)
- t = idf
- p = idf probabiliste

normalisation

- n = pas de normalisation (=1)
- c = normalisation cosinus
- u = normalisation à pivot

Combinaison des pondérations

calcul en fonction du schéma de pondération :
avec le schéma ABC.DEF

$$s(d,q)=\frac{\sum_{i=1}^{n}(A\cdot D)\times(B\cdot E)}{C\cdot F}$$

Combinaison des pondérations

exemple: un schéma populaire: Lnu.ltc

$$s(d,q)=\frac{\sum_{i\ t.q.\ t_i\in d\cap q}\left(\frac{1+\log(tf_i(d))}{1+\log(\text{avg}_j tf_j(d))}\right)\times(1+\log(tf_i(q))\times idf_i(q))}{(1-slope)\times pivot+slope\times NT(d)\cdot\sqrt{\sum_{i=1}^{n} freq\,(1+\log(tf_i))\times idf_i}}$$

avec $tf_i(d)=freq(t_i,d)$

$$idf_i(d)=1+\log\left(\frac{N}{df(t_i)}\right)$$

$NT(d)=$ *nombre de termes différents dans d*

Exemple d'évaluation de différentes pondérations

Evaluation de différents modèles

(extrait de *Report on CLEF-2005 Evaluation Campaign,* Jacques Savoy and Pierre-Yves Berger)

	Mean average precision					
Query Model \ # of queries	French T 50 queries	French TD 50 queries	French TDN 50 queries	Portuguese T 50 queries	Portuguese TD 50 queries	Portuguese TDN 50 queries
Prosit	0.2895	0.3696	**0.3961**	0.2755	0.3438	0.3697
doc=Okapi, query=npn	**0.3029**	**0.3754**	0.3948	**0.2873**	**0.3477**	**0.3719**
doc=Lnu, query=ltc	0.2821	0.3437	0.3703	0.2611	0.3338	0.3517
doc=dtu, query=dtn	0.2726	0.3365	0.3633	0.2571	0.3221	0.3338
doc=atn, query=ntc	0.2809	0.3328	0.3507	0.2458	0.3076	0.3433
doc=ltn, query=ntc	0.2588	0.3066	0.3232	0.2149	0.2535	0.2740
doc=ntc, query=ntc	0.1862	0.2175	0.2335	0.1553	0.1868	0.2221
doc=ltc, query=ltc	0.1916	0.2363	0.2611	0.1625	0.2234	0.2543
doc=lnc, query=ltc	0.2050	0.2616	0.2953	0.1811	0.2475	0.2950
doc=bnn, query=bnn	0.1153	0.0937	0.0514	0.1309	0.1322	0.0900
doc=nnn, query=nnn	0.1148	0.0987	0.0748	0.0630	0.0639	0.0453

Table 3: Mean average precision of various single searching strategies (French & Portuguese languages)

Combinaison des pondérations

Il n'y a pas de schéma de pondération optimal

dépend du corpus et de la requête

le choix du schéma se fait empiriquement en fonction des évaluations

Autres mesures de similarité

mesures dérivées des mesures ensemblistes

mesure de Dice

$$dice(q,d)=\frac{2\times\sum_{i=1}^{n} q_i\cdot d_i}{\sum_{i=1}^{n} q_i^2+\sum_{i=1}^{n} d_i^2}$$

mesure de Jaccard

$$jaccard(q,d)=\frac{\sum_{i=1}^{n} q_i\cdot d_i}{\sum_{i=1}^{T} q_i^2+\sum_{i=1}^{n} d_i^2-\sum_{i=1}^{n} q_i\cdot d_i}$$

Autres mesures de similarité

si les vecteurs sont normalisés par la norme L1, ils peuvent être considérés comme des distributions de probabilité

➜ mesures entre distributions de probabilité

distance du x^2

$$s_{\chi^2}(d,q)=\sqrt{\sum_{i=1}^{n}\frac{(d_i-q_i)^2}{\rho_i}}$$

où $\rho_i=\frac{\sum_d d_i}{N}$ *est la distribution marginale sur la collection*

- Particularité : mesure sensible à la différence hors intersection
- deux termes de même profil peuvent être considérés comme le même terme sans changer les valeurs des distances

Autres mesures de similarité

autres mesures dérivées de la théorie de l'information:

divergence de Jensen-Shannon

$$JS(d\|q) = D\left(d\left\|\frac{d+q}{2}\right.\right) + D\left(q\left\|\frac{d+q}{2}\right.\right)$$

avec D(x||y) la divergence de Kullback-Leibler

$$D(x\|y) = \sum_{i=1}^{n} x_i \cdot \log \frac{x_i}{y_i}$$

- test statistique de l'hypothèse que *d* et *q* sont deux observations empiriques de la même distribution théorique

Autres mesures de similarité

La très grande majorité des systèmes utilisent la mesure du cosinus

les propriétés théoriques des mesures utilisées nécessaires pour la recherche d'information ont peu été étudiées:

- en général, les similarités utilisées ne dépendent que de l'intersection (termes communs)
- les vecteurs sont normalisés (prise en compte de la longueur des documents)

Utilisation de l'index dans un modèle vectoriel

lorsqu'on utilise une mesure ne dépendant que de l'intersection, on ne prend en compte que les termes appartenant à cette intersection.

Algorithme :

$s(d_j , q) \leftarrow 0$ pour tout les documents d_j de D

pour chaque terme t_i de la requête (avec un poids q_i)

récupérer l'ensemble D_i des documents d_{ij} tels que $t_i \in d_{ij}$

pour chaque d_{ij} de D_i

p_{ij} est le poids du terme t_i dans d_{ij}

$s(d_{ij} , q) \leftarrow s(d_{ij} , q) + p_{ij} \times q_i$

Extensions du modèle vectoriel

Retour de pertinence (*relevance feedback*)

Expansion de requêtes

- thesaurus
- intégration de co-occurrences

modèle d'analyse sémantique latente
(*LSI -- Latent Semantic Indexing*)

Retour de pertinence

réaliser une première recherche à l'aide des termes de la requête

enrichir la requête en utilisant l'information de pertinence *fournie par l'utilisateur* sur les premiers documents résultats

modèle d'expansion de Rocchio:

$$q_{new} = \alpha \times q_{orig} + \beta \times \frac{1}{|R|} \sum_{i=1}^{n} d_i - \gamma \times \frac{1}{\bar{R}} \sum_{i=1}^{n} d_i$$

R est l'ensemble des vecteurs des documents pertinents

$\bar{R}$ est l'ensemble des vecteurs des documents non pertinents

pseudo relevance-feedback

sans intervention de l'utilisateur

on suppose que les *n* premiers documents retournés par le système sont pertinents

on ne fait pas d'hypothèse sur la non-pertinence des autres documents (expansion positive seulement):

$$q_{new} = \alpha \times q_{orig} + \beta \times \frac{1}{|R|} \sum_{i=1}^{n} d_i$$

en général, le retour de pertinence améliore les résultats globaux, mais l'amélioration dépend fortement des requêtes (*e.g.* si la collection ne contient que peu de documents pertinents)

Expansion de requêtes

Expansion de requêtes en utilisant des thesaurus

- synonymes
- Hyponymes *(est-un)*
- Hypéronymes *(est-un)*
- Méronymes *(fait partie de)*

- mots proches dans un thésaurus (partageant des liens)

thesaurus le plus utilisé : *WordNet*

Expansion de requêtes

similarité entre termes fondée sur les co-occurrences:

similarité cosinus

$$s_{\cos}(t,t') = \frac{df(t,t')}{\sqrt{df(t) \times df(t')}}$$

information mutuelle

$$IM(t,t') = \log \frac{P(t,t')}{P(t)P(t')} = \log \frac{freq(t,t')}{freq(t) \times freq(t')}$$

Expansion de requêtes

expansion des requêtes par des termes similaires, au sens de ces similarités

résultats mitigés

Souvent des termes de même fréqence

- Pas de différence dans les mesures de similarité

Intégration de co-occurrences

Similarité entre termes fondée sur les co-occurrences partagées:

> les termes qui apparaissent dans les mêmes contextes sont similaires: la sémantique d'un mot est reliée à l'ensemble des contextes dans lesquels il apparaît (sémantique distributionnelle)

certains X, par exemple, attaquent naturellement les rats. (C. Darwin)

quelque X sur les toits, marchant lentement, bombait son dos aux rayons pâles du soleil. (G. Flaubert)

il entendait au loin dans la forêt les miaulement des X. (A. France)

Intégration de co-occurrences

Contexte = l'ensemble des mots qui apparaissent *autour* du mot considéré:

c(X)={attaquer,rat,toit,marcher,bomber,dos,miaulement...}

profils de co-occurrence: vecteurs de fréquences de co-occurrence

$$c(t_i)=(c_{i1},\ldots,c_{in})$$

avec c_{ij} = *fréquence de co-occurrence de* t_i *et* t_j

similarité entre termes = similarité entre profils de co-occurence (par exemple, cosinus)

Intégration de co-occurrences

pour l'expansion de requête

➥ ajouter à la requête des termes dont les profils de co-occurrences sont similaires

modèle DSIR (*Distributional Semantics Information Retrieval*)

représentation de la requête et des documents par la moyenne des profils de co-occurrence des termes qui les composent

$$d_i = \sum_{j=1}^{n} c_{ij}\, w(t_j)$$

ou modèle hybride

$$d_i = \alpha\, w(t_i) + (1-\alpha) \times \sum_{j=1}^{n} c_{ij}\, w(t_j)$$

Intégration de co-occurrences

Du point de vue matriciel, on peut définir une matrice de co-occurrence dont chaque ligne est le profil de co-occurrence d'un terme

$$C = \begin{pmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{T1} & \cdots & c_{Tn} \end{pmatrix}$$

NB: la matrice n'est pas forcément carrée. Ce modèle peut aussi être utilisé pour réduire la dimension de l'espace de représentation tout en prenant en compte plus de mots.

la matrice de représentation des documents s'écrit alors comme un produit matriciel

$$D' = D_T C$$

$$D' = \alpha D_n + (1-\alpha) D_n C$$

Avec : *T le nombre de termes du vocabulaire*
N le nombre de termes pour l'indexation
D_T *la matrice documents ×termes sur T termes*
D_n *la matrice documents ×termes sur n termes*

Latent Semantic Indexing

« *tente de prendre en compte la structure sémantique des termes, potentiellement implicite, représentée par leurs dépendances cachées.* »

réduction de la dimensionnalité de l'espace vectoriel de représentation en gardant le plus d'information possible.

Latent Semantic Indexing

Décomposition en valeurs singulières (SVD) sur la matrice documents ×termes D

$$D = U \Sigma V^t$$

avec

U et V orthogonormées ($UU^t=1$, $VV^t=1$)

Σ matrice diagonale de dimension *m* égale au rang de la matrice D, dont les valeurs sur la diagonale sont ordonnées ($\sigma_{11} > \sigma_{22} > ... > \sigma_{\mu\mu}$)

Latent Semantic Indexing

si *N* est la taille de la collection, *n* la taille du lexique et m le rang de la matrice *D*

$$\begin{pmatrix} d_{11} & \cdots & \cdots & d_{1N} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ d_{n1} & \cdots & \cdots & d_{nN} \end{pmatrix} = \begin{pmatrix} u_{11} & \cdots & u_{1m} \\ \vdots & & \vdots \\ \vdots & \ddots & \vdots \\ u_{n1} & \cdots & u_{nm} \end{pmatrix} \begin{pmatrix} \sigma_{11} & & 0 \\ & \ddots & \\ 0 & & \sigma_{mm} \end{pmatrix} \begin{pmatrix} v_{11} & \cdots & \cdots & v_{1N} \\ \vdots & & \ddots & \vdots \\ v_{m1} & \cdots & \cdots & v_{mN} \end{pmatrix}$$

Latent Semantic Indexing

L'approche LSI consiste à réduire la dimension de l'espace de représentation de façon à garder le plus d'information possible

supprimer les dernières valeurs singulières

$\sigma_{ii}=0$ pour $i>k$ ($k<m$)

on réduit ainsi la dimension de Σ à k

on obtient un approximation de D (meilleure approximation de D de rang k au sens des moindres carrés)

$$\hat{D}=\hat{U}\,\hat{\Sigma}\,\hat{V}^{t}$$

Latent Semantic Indexing

$\hat{V}\,\hat{\Sigma}$ permet de plonger les documents dans un espace vectoriel de dimensions *k*

la distance entre les documents deux à deux se calcule par

$$\hat{D}\,\hat{D}^t = \hat{V}\,\hat{\Sigma}^2\,\hat{V}^t$$

la distance entre une requête et l'ensemble de document se fait par la transformation de la requête q (vecteur des termes) en un pseudo-document

$$D_q = q\,\hat{T}\,\hat{\Sigma}^{-1}$$

qu'on ajoute comme une ligne de la matrice D, et on recalcule sa distance par rapport aux autres documents

Latent Semantic Indexing

symétriquement, les termes peuvent être représentés également par un vecteur dans l'espace de dimension *k* (représentatif de leur distribution dans la collection)
➥ permet de dériver une représentation des termes sur laquelle on peut faire un calcul de similarité

modèle utilisé pour de nombreuses applications (filtrage d'information, recherche crosslingue)

sur des corpus moyens ou petits, ce modèle a montré de bonnes performances (moins de différences sur des gros corpus)

Latent Semantic Indexing

inconvénients:

- la valeur de *k* doit être fixée empiriquement (300/500 semble donner de bons résultats)
- les dimensions de l'espace de représentation ne sont pas interprétables