Bases de données documentaires et distribuées, http://b3d.bdpedia.fr

Principes de la Recherche d'Information

Information Retrieval, définition

Une définition

La **Recherche d'Information** (*Information Retrieval*, IR) consiste à trouver des **documents** peu ou faiblement structurés, dans une grande **collection**, en fonction d'un **besoin d'information**.

Recherche sur le Web. Utilisée quotidiennement par des milliards d'utilisateurs.

Recherche dans votre boîte mail.

Recherche sur votre ordinateur (*Spotlight*).

Recherche dans une base documentaire, publique ou privée.

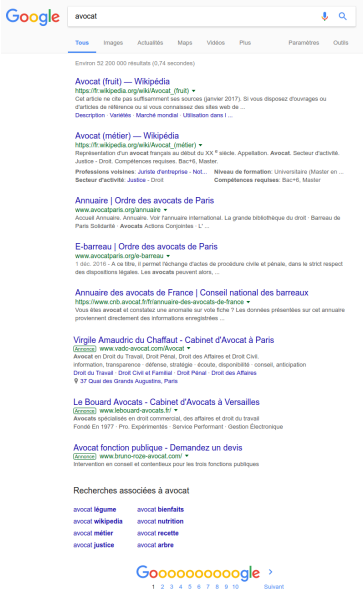
Information Retrieval, définition

Recherche plein texte : on cherche à examiner tous les mots de chaque document enregistré et à essayer de les faire correspondre à ceux fournis par l'utilisateur

À bien distinguer d'une recherche type "base de données" : requête structurée, données structurées, réponse « exacte ».

Contrairement à une base de données (SQL), le résultat dépend de **l'interprétation** d'un besoin. On ne peut jamais dire qu'un résultat est totalement exact (ou totalement faux).

Évaluation d'un moteur de recherche



Évaluation d'un moteur de recherche

Évaluation d'un moteur de recherche

Deux notions importantes

Faux positifs : ce sont les documents **non pertinents** inclus dans le résultat ; ils ont été sélectionnés à tort.

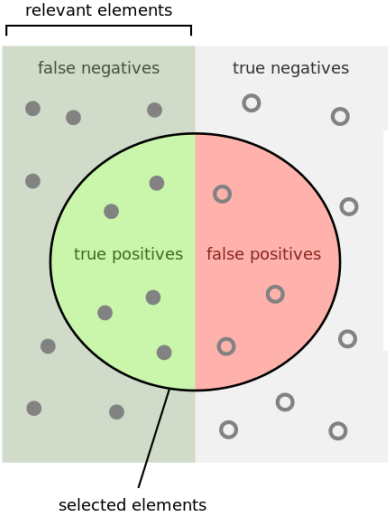
Faux négatifs : ce sont les documents **pertinents** qui **ne sont pas** inclus dans le résultat.

La recherche plein texte est susceptible de récupérer beaucoup de faux positifs.

La récupération de documents non pertinents est souvent provoquée par l'ambiguïté inhérente au langage naturel ;

En général, chercher à réduire les faux positifs entraîne l'augmentation des faux négatifs, et réciproquement.

Faux-positifs et négatifs, précision et rappel



How many selected items are relevant?

Precision =

How many relevant items are selected?

Recall =

Exemple de base

Un ensemble (modeste) de documents nous servira de guide.

d_1 Le loup est dans la bergerie.
d_2 Le loup et le trois petits cochons.
d_3 Les moutons sont dans la bergerie.
d_4 Spider Cochon, Spider Cochon, il peut marcher au plafond.
d_5 Un loup a mangé un mouton, les autres loups sont restés dans la bergerie.
d_6 Il y a trois moutons dans le pré, et un mouton dans la gueule du loup.
d_7 Le cochon est à 12 Euros le Kg, le mouton à 10 Euros/Kg.
d_8 Les trois petits loups et le grand méchant cochon.

Le besoin et la solution

Besoin

On veut chercher tous les documents parlant de loups, de moutons mais pas de bergerie.

Parcourir tous les documents ? (`grep`)

- potentiellement long ;
- critère "pas de bergerie" n'est pas facile à traiter ;
- autres types de recherche ('loup' doit être **près** de 'mouton'") sont difficiles ;
- comment **classer** par pertinence les documents trouvés ?

Structure spécialisée : la **matrice d'incidence** et surtout son inversion.

Matrice avec documents en ligne

On sélectionne un ensemble de mots (ou **termes**), constituant notre **vocabulaire**

La matrice d'incidence

	loup	**mouton**	**cochon**	**bergerie**	**pré**	**gueule**
d_1	1	0	0	1	0	0
d_2	1	0	1	0	0	0
d_3	0	1	0	1	0	0
d_4	0	0	1	0	0	0
d_5	1	1	0	1	0	0
d_6	1	1	0	0	1	1
d_7	0	1	1	0	0	0
d_8	1	0	1	0	0	0

Documents en ligne, termes en colonnes. Dans chaque cellule :
1 si le terme est dans le document, 0 sinon.

Recherche (loup, mouton, et pas bergerie)

On prend les vecteurs binaires des **termes** (les colonnes).

Loup : 11001101

Mouton : 00101110

Bergerie : 01010011

Puis :

ET logique sur les vecteurs de Loup et Mouton, on obtient 00001100.

ET logique avec le **complément** du vecteur de Bergerie (01010111)

On obtient 00000100, d'où on déduit que la réponse est limitée au document d_6.
Opération binaires **très efficace**, mais...

Passons à grande échelle

Un million de documents, mille mots chacun en moyenne.

Disons 6 octets par mot, soit 6 Go (ce n'est pas une si grosse base que cela)

Disons 500 000 termes **distincts**

$\Rightarrow$ la matrice d'incidence a :

10^6 lignes et 500 000 colonnes soit 500×10^9 bits

soit 62 Go approximativement

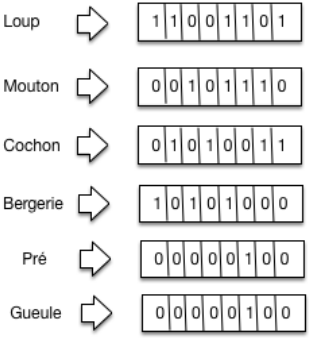
Ne tient pas en mémoire, ce qui va beaucoup compliquer les choses....

Comment faire mieux ?

On peut faire mieux

Il vaut mieux avoir les termes en ligne pour disposer des vecteurs dans une zone mémoire contigue.

On parle de matrice inversée, et de **liste inversée**. Une liste par terme ; dans chaque liste, 1 pour les documents contenant le terme.

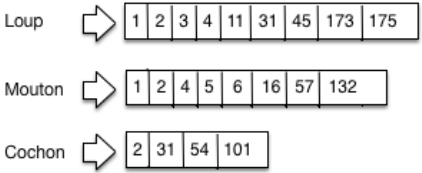


Important

Mises à jour beaucoup plus difficiles car elles impliquent la réorganisation d'une liste compacte. Prix à payer pour l'efficacité en **lecture** (recherche).

On peut encore faire mieux

La matrice est **creuse** : il n'y a que 10^9 positions avec des 1, soit un sur 500.



Essentiel

On place dans les cellules **l'identifiant** du document. De plus chaque liste est **triée** sur l'identifiant du document.

Index inversé

La structure utilisée dans **tous** les moteurs de recherche.

- Un **répertoire** contient tous les **termes**.
- Une **liste** (inversée) est associée à chaque **terme**, **triée** par docId.
- Chaque élément de la liste est appelé une **entrée**.

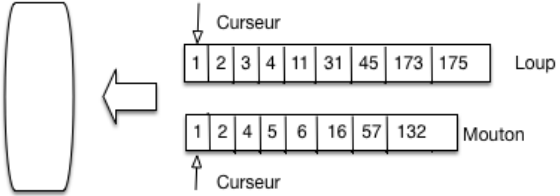
Efficacité : le répertoire devrait toujours être en mémoire ; les listes, autant que possible en mémoire, sinon fichiers contigus sur le disque.

Documents parlant de loup ET de mouton

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.

On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !

On avance sur la liste du plus petit docId.

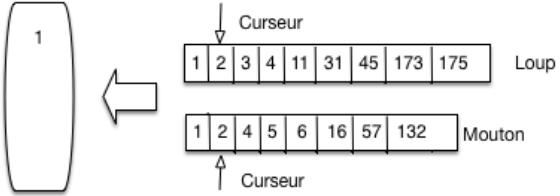


Documents parlant de loup ET de mouton

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.

On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !

On avance sur la liste du plus petit docId.



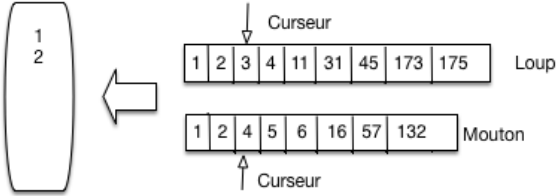
C'est une recherche dite **Booléenne** : pas de classement, résultat exact. On peut faire mieux...

Documents parlant de loup ET de mouton

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.

On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !

On avance sur la liste du plus petit docId.



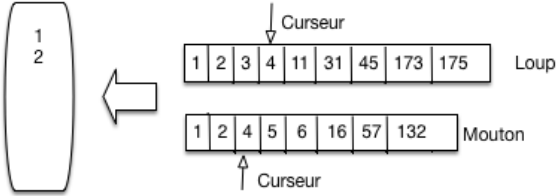
C'est une recherche dite **Booléenne** : pas de classement, résultat exact. On peut faire mieux...

Documents parlant de loup ET de mouton

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.

On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !

On avance sur la liste du plus petit docId.



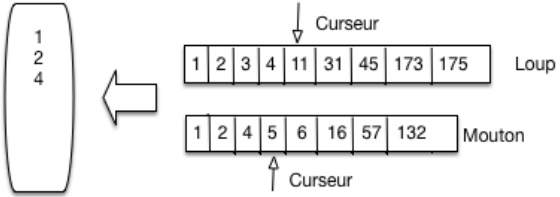
C'est une recherche dite **Booléenne** : pas de classement, résultat exact. On peut faire mieux...

Documents parlant de loup ET de mouton

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.

On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !

On avance sur la liste du plus petit docId.



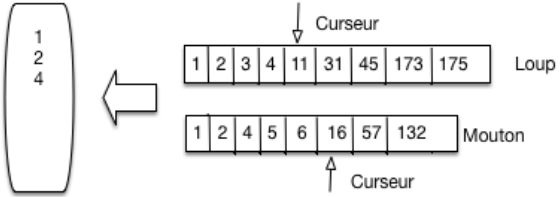
C'est une recherche dite **Booléenne** : pas de classement, résultat exact. On peut faire mieux...

Documents parlant de loup ET de mouton

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.

On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !

On avance sur la liste du plus petit docId.



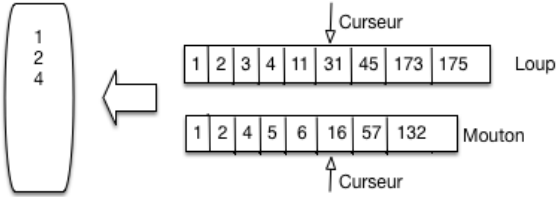
C'est une recherche dite **Booléenne** : pas de classement, résultat exact. On peut faire mieux...

Documents parlant de loup ET de mouton

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.

On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !

On avance sur la liste du plus petit docId.



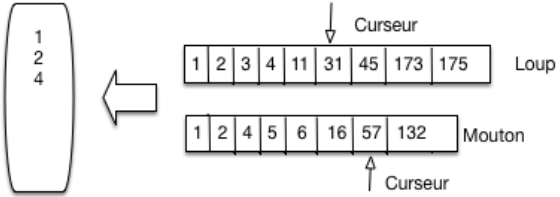
C'est une recherche dite **Booléenne** : pas de classement, résultat exact. On peut faire mieux...

Documents parlant de loup ET de mouton

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.

On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !

On avance sur la liste du plus petit docId.



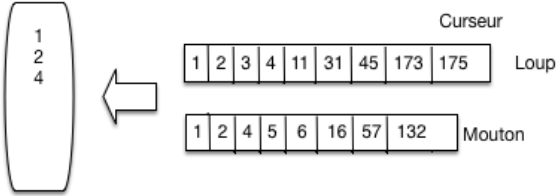
C'est une recherche dite **Booléenne** : pas de classement, résultat exact. On peut faire mieux...

Documents parlant de loup ET de mouton

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.

On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !

On avance sur la liste du plus petit docId.



C'est une recherche dite **Booléenne** : pas de classement, résultat exact. On peut faire mieux...

Premier bilan

Fondements techniques : index inversé, tri et compaction, parcours linéaire.

Permet d'effectuer des recherches **puissantes** et **flexibles**.

Garantissent une très grande efficacité.

Quels termes ? Quels termes indexe-t-on ? Beaucoup moins facile que ça n'en a l'air...

Quelles requêtes ? De la plus simple ("sac de mots") à plus structurée

Performance. compression, distribution, optimisation des accès, mises à jour, etc.

Classement ? comment classer en premier les documents pertinents ?

La suite dans les prochaines sessions