Aperçu du projet "entier"

TP1 collecteur → le dictionnaire + index inversé (relation mots-pages)

mot m → page 3 (TF(m,d)) page 8 (TF) ... ← pages où m apparaît

IDF(m)

+ graphe des pages dans forme CLI

1er score des pages selon la fréquence des mots :

* pour chaque mot m, $IDF(m)$.
* pour chaque page d, chaque mot m dans la relation mots-pages : $TF(m,d)$ normalisé (TF/N_d)

$= \sum_{m \in d} TF(m,d)^2$

tronquer les listes

TP2 (pagerank) : calcul un score pour chaque page d indépendamment des mots contenus dans d (on regarde le graphe)

→ pr(d)

page

TP 3 traitement de la requête.

Req = m_1 m_2 ... m_k.

1. Trouver les pages contenant tous les mots → P.
2. Calculer le vecteur normalisé de la requête (dim. k, base ($IDF(m_1)$ ... $IDF(m_k)$))
3. pour chaque page $d \in P$, on calcule le score de la page pour la requête :
   - produit scalaire du vecteur de la page (restreint à m_1 ... m_k) : $TF(m_1,d)$ .. $TF(m_k,d)$
   - avec le vecteur de la requête (normalisé)

   → score s_1.
4. Pour chaque page $d \in P$, calculer $S_d = \alpha s_1 + \beta pr(d)$.

score de d pour la requête

produit scalaire → $\sum_{i=1}^{k} IDF(m_i) \times TF(m_i,d)$

si $IDF(m_2) \times TF(m_2,d) \simeq 10^{-6}$ (?) très faible → contribue peu au score → on peut supprimer la page d de la liste de m_2

On enrobe tout ça dans un site web.

client —requête→ serveur

serveur → score

← liste des pages triées