Formulaire - MAC (RI)

Leonard Cseres | December 17, 2025

Modèle Booléen Modèle le plus simple pour fonder un système RI. Requêtes en expressions booléennes (AND, OR, NOT). Résultats satisfont exactement l'expression: soit correspond, soit pas.
+ Bon pour utilisateurs experts et applications
+ Précis et simple
− Difficile pour majorité des utilisateurs
− Problème "feast or famine": trop peu (0) ou beaucoup trop (milliers) de résultats

Matrice d'incidence terme-document Solution simple: entrée = 1 si terme apparaît dans document, 0 sinon. Opérations booléennes bit à bit sur vecteurs.

Problème: pour collections grandes ($N = 10^6$ docs, $M = 500000$ termes), matrice de $M \times N$ est trop grande et extrêmement creuse.

Index Inversé Structure clé de la RI. Pour chaque terme t, stocke liste de tous documents qui contiennent t.
- DocID: numéro de série du document
- Liste d'occurrences (postings): liste de DocIDs
- Dictionnaire: tous les termes

Construction d'index inversé
1. Tokenisation: découper séquence de caractères en tokens
2. Modules linguistiques: normalisation, stemming, stop words
3. Indexation:
   - Séquence de paires (token modifié, DocID)
   - Trier par terme puis DocID
   - Fusionner entrées multiples
   - Ajouter fréquence documentaire (df)

Traitement requêtes AND
- Récupérer listes d'occurrences pour Brutus et Calpurnia
- Fusionner (intersection) les deux listes triées
- Complexité: $O(x + y)$ où $x, y =$ longueurs des listes

Crucial: listes doivent être triées.

Index positionnel Stocke position(s) exacte(s) où chaque terme apparaît dans document.

Format: `<terme, nb docs; doc1: pos1, pos2, ...; doc2: pos1, pos2, ...; ...>`
+ Supporte requêtes phrases (ex: "Applied Science University")
+ Supporte requêtes de proximité (ex: Bank /3 scandal)
− Taille: 2-4× plus grand qu'index non-positionnel
− 35-50% du volume texte original

Modèle Vectoriel Introduit recherche avec classement. Retourne documents ordonnés par probabilité d'utilité. Score $\in [0, 1]$ pour chaque paire (requête, document).

Motivation: résoudre problème "feast or famine" du modèle booléen.

Coefficient Jaccard $\text{JACCARD}(A, B) = \frac{|A \cap B|}{|A \cup B|}$
− Ne tient pas compte fréquence des termes
− Ne distingue pas termes spécifiques vs. génériques
+ Normalisation simple

Fréquence de terme (tf) $tf_{t,d}$ = nombre de fois où terme t apparaît dans document d.

Pondération log-frequency: $w_{t,d} = \begin{cases} 1+\log_{10}(tf_{t,d}) \text{ si } tf_{t,d}>0 \\ 0 \text{ sinon} \end{cases}$

Pertinence n'augmente pas proportionnellement à fréquence.

Fréquence documentaire (df) et idf df_t = nombre de documents contenant terme t.
- df_t élevé $\Rightarrow$ terme t général
- df_t bas $\Rightarrow$ terme t spécifique (rare)

Poids idf (inverse document frequency): $idf_t = \log_{10}(N/df_t)$

où N = nombre total de documents.

Pondération tf-idf Produit du poids tf et poids idf: $w_{t,d} = \big(1 + \log(tf_{t,d})\big) \times \log_{10}(N/df_t)$

Meilleur schéma de pondération pour RI:
↑ Augmente avec occurrences dans document
↑ Augmente avec rareté du terme

Représentation vectorielle Documents et requêtes = vecteurs dans espace $|V|$-dimensionnel.
- $\vec{d}_j = \big(w_{1j}, w_{2j}, ..., w_{tj}\big)$
- $\vec{q}_j = \big(w_{1j}, w_{2j}, ..., w_{tj}\big)$

Modèle du sac de mots: ordre des mots ignoré.

Similarité cosinus Distance euclidienne inadaptée (pénalise vecteurs de longueurs différentes).

Classer par angle entre requête et document: $\text{Sim}(q, d) = \cos(q, d) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}|\,|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} d_i^2}}$

Normalisation de longueur Vecteur normalisé = diviser chaque composante par sa longueur.

Norme L2: $\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$

Mappe vecteurs sur sphère unitaire. Documents longs et courts ont poids du même ordre. Similarité = simple produit scalaire.

Tokenisation Découper texte en tokens (séquences de caractères). Chaque token = candidat pour entrée d'index.

Défis: John's, state-of-the-art, U.S.A., etc.

Normalisation des termes Mapper texte de document et termes de requête à même forme.
- Accents, minuscule/majuscule
- Cohérence entre documents et requêtes essentielle

Stop words (mots vides) Termes très courants avec peu de valeur sémantique. Représentent 30% des postings pour 30 premiers mots.

Suppression avec stop list:
+ Économie espace index
+ Traitement requêtes plus rapide
− Problématique pour requêtes phrases
− Problématique pour relations ("flights to London" vs. "flights from London")

Tendance actuelle: garder stop words.

Stemming (racinisation) Processus heuristique qui coupe extrémités des mots pour réduire à forme de base. Suppression affixes dérivationnels.
+ Réduit taille index
+ Améliore rappel
− Peut réduire précision
Dépend de la langue
Si appliqué lors indexation, doit être appliqué aux requêtes

Lemmatisation Supprime fins flexionnelles, retourne forme de base (lemme) en utilisant vocabulaire et analyse morphologique.

Plus précis que stemming, mais plus complexe.

Évaluation des systèmes RI Comparer efficacité de différentes techniques. Collection de tests requiert:
1. Collection de documents de référence
2. Suite de requêtes de référence
3. Jugements de pertinence pour chaque paire (requête, document)

Collections publiques: CACM, AP, GOV2

Précision et Rappel Mesures de base (ensembles non ordonnés):

Précision $= \frac{\text{docs pertinents retrouvés}}{\text{total docs retrouvés}} = t\frac{p}{tp+fp}$

Rappel $= \frac{\text{docs pertinents retrouvés}}{\text{total docs pertinents}} = t\frac{p}{tp+fn}$

Inversement proportionnels. Rappel ne diminue jamais quand plus de docs retournés.

Mesure F Moyenne harmonique pondérée de précision et rappel: $F_\beta = \frac{(\beta^2+1) \cdot P \cdot R}{\beta^2 \cdot P + R}$

F_1 (moyenne harmonique): $F_1 = \frac{2 \cdot P \cdot R}{P+R}$

Moyenne harmonique souligne importance petites valeurs.

Accuracy $\frac{tp+tn}{tp+fp+fn+tn}$

Inadaptée pour RI: >99.9% documents non pertinents. Précision et rappel se concentrent sur vrais positifs.

Average Precision (AP) Pour résultats classés. Moyenne des valeurs de précision aux positions où document pertinent récupéré:

$\text{AP} = \frac{\sum_{k=1}^{n} (\text{Precision}(k) \times \text{rel}(k))}{\text{nombre total docs pertinents}}$

où $\text{rel}(k) = 1$ si document à rang k est pertinent, 0 sinon.

Mean Average Precision (MAP) Moyenne des AP sur plusieurs requêtes: $\text{MAP} = \frac{\sum_{q=1}^{Q} \text{AP}(q)}{Q}$

Mesure standard pour comparer performance globale de systèmes RI.

Graphique Rappel-Précision Courbe avec valeurs rappel et précision pour chaque document retourné.

Précision interpolée: précision à niveau de rappel r = précision max observée pour tout niveau de rappel $r' \geq r$.

Précision moyenne interpolée à onze points: calculer précision interpolée pour 11 niveaux de rappel standard (0.0, 0.1, 0.2, …, 1.0), puis moyenner sur plusieurs requêtes.

Précision @ rang K (P@K) Précision calculée à position fixe K (typiquement 5, 10, 20).

Facile à calculer, utile quand seuls top résultats importent. Insensible aux positions $< K$.

R-Précision Précision à la R-ième position, où $R =$ nombre total de documents pertinents pour requête.

Adapte mesure au nombre de docs pertinents par requête.

Implémentation pratique Documents ne contenant aucun mot-clé de requête n'affectent pas classement.

Pipeline:
1. Prétraitement: tokenisation, stop words, stemming
2. Indexation: construire index inversé avec structures rapides (B-trees, hashtables)
3. Recherche: utiliser index pour trouver docs avec $\geq$ 1 mot de requête
4. Classement: calculer scores cosinus, trier par score décroissant

Accumulation term-at-a-time: calculer scores en ajoutant contributions des termes de requête, un à un.

Schémas de pondération tf-idf Trois composantes: term frequency, document frequency, normalization.

Schéma standard:
- Document: l-n-c (tf log, pas d'idf, norm cosinus)
- Requête: l-t-c (tf log, idf, norm cosinus)

Term frequency:
- n (natural): $tf_{t,d}$
- l (logarithm): $1 + \log tf_{t,d}$
- a (augmented): $0.5 + \frac{0.5 \cdot tf_{t,d}}{\max_t tf_{t,d}}$
- b (boolean): $\begin{cases} 1 \text{ si } tf_{t,d}>0 \\ 0 \text{ sinon} \end{cases}$
- L (log ave): $\frac{1+\log(tf_{t,d})}{1+\log(\text{ave}_{t \in d}(tf_{t,d}))}$

Document frequency:
- n (no): 1
- t (idf): $\log(N/df_t)$
- p (prob idf): $\max\{0, \log((N - df_t)/df_t)\}$

Normalization:
- n (none): 1
- c (cosine): $\frac{1}{\sqrt{w_1^2+w_2^2+...+w_M^2}}$
- u (pivoted unique): $1/u$
- b (byte size): $1/\text{CharLength}^\alpha, \alpha < 1$

Complexité indexation Indexation de m documents de n tokens chacun: $O(mn)$.

Complexité = simple lecture du corpus.

Avantages modèle vectoriel
+ Simple et mathématique
+ Prend en compte tf (local) et idf (global)
+ Correspondance partielle et résultats classés
+ Fonctionne bien en pratique
+ Implémentation efficace pour grandes collections

Limitations modèle vectoriel
− Manque informations syntaxiques (structure phrase, ordre mots)
− Manque interprétation sémantique (sens des mots)
− Hypothèse indépendance termes (ignore synonymie)
− Pas de contrôle booléen (exiger terme dans document)