Fiche RI

Charles Vin

2023

1 Généralité

- RI ad-hoc : c'est ce qu'on fait : trouver parmi un ensemble d'articles ceux qui concernent un sujet spécifique.
- Indexation = encodage des documents avec un modèle RI
- Deux types d'index
  - Index normal : Document : (terme, nombre/score)
  - Index inversé : Mot : (document, nombre/score)
- Requete sur index : on somme les scores pour obtenir le score final d'un document
- Stemming : ne garde que la racine des mots, un peu moche
- Lematization : retour vers un mot complet
- Strategie de recherche :
  - Problème : On a beaucoup de doc, on cherche les K premiers
  - Deux strat pour index inversé :
    - TAAT : traiter les terme un par un, fonctionne bien sur des petits corpus où il y a une grande différence de score
      1. Trier l'index de chaque terme par score croissant -> une grosse liste ordonnée terme puis score
      2. Parcourir par terme et maintenir un dictionnaire avec les scores par document, optimisation : utilisation d'une Heap
    - DAAT : traiter les doc un par un, plus efficace pour les grandes collections, plus

2 Loi de Zipf

Stipule que la fréquence d'occurrence d'un mot est inversement proportionnelle à celle de son rang dans la liste des mots classés par fréquence. Les mot les plus fréquent sont beaucoups plus fréquents que moins fréquent. Le 1er mot est environ 2 fois plus fréquent que le 2nd qui est 2 fois plus fréquent que le 3e etc..

$$\frac{\frac{1}{r^s}}{\sum_{n=1}^{N} \frac{1}{N}} \approx \frac{1}{r^s}, frequence = \frac{\lambda}{rank}.$$

Avec r le rang, N la taille du corpus et s un paramètre spécifique au corpus.

3 Loi de Heaps

Lien entre le nombre de mots distinct et le nombre de mots :
- les nouveaux mots apparaissent moins fréquenmment quand le vocabulaire croît.
- La taille du vocabulaire n'a pas de borne supérieure (nom propres, erreur de typo)

$$V = Kn^{\beta}.$$

Avec V taille du vocabulaire, N taille du texte, K, β paramètre spécifique du texte.

4 TF-IDF

— Term Frequency : Une pondération locale, fréquence du terme dans le document $\frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$
— Inverse Document frequency : Une pondération globale, fréquence inverse décroit vers 0 si le terme apparait dans tous les documents $\log \frac{N}{df(t_i)}$ avec df nombre de documents contenant le terme, N nombre de document.
— TF-IDF : $tf*idf$, il existe plein de variance. En particulier en TD on a fait un "count idf" : $f_{t,d} \log \frac{N}{n_t}, n_t =$ nombre de document où t est présent.

5 Modèle booléen

formule logique into score binaire, pas de pondération rien

6 Modèle vectoriel

Score de distance entre deux vecteurs : un de document score (classiquement un tf) et celui de la requète (classiquement binaire par terme). Inner product $< X, Y >$, cosinus $\frac{<X,Y>}{\|X\|\|Y\|}$

7 Modèle probabiliste

Soit R une v.a.r binaire pour un dociment d est pertinent pour la question q.
— On cherche la proba $P(R|q,d)$ que le document soit pertinent pour la question et le document.
— Par bayes on tombe sur $P(d|R,q)P(R|q)$.
— Un document se décompose en terme **indépendant** de cette manière d : $(\bigwedge_{t \in d}) \wedge (\bigwedge_{t \notin d} t \notin d)$
— $\prod_{t \in d} P(t \in d|R,q) \prod_{t \notin d} P(t \notin d|R,q) P(R|q)$
— Et on peut estimer la proba qu'un terme apparaisse dans un document pertinent $p_t = P(t \in d|R,q), 1 - p_t = P(t \notin d|R,q)$.
— Finalement on peut développer un peu avec un tricks sur le produit et virer les constantes

$$\begin{aligned} P(R|d,q) &= P(R|q) \prod_{t \in d} p_t \prod_{t \notin d} 1 - p_t \\ &= P(R|q) \prod_{t \in d} p_t \frac{1}{1-p_t} \prod_{t \in \mathcal{T}} 1 - p_t \\ &\propto \prod_{t \in d} p_t \frac{1}{1-p_t} \end{aligned}$$

— De base le model pose un rapport de vraissemblance pour avoir un score $\frac{P(R|q,d)}{P(\not R|q,d)}$, même développement qu'au dessus, puis passage au log.

$$s(q,d) = \sum_{t \in q \sqcap d} \log \frac{p_t(1-u_t)}{u_t(1-p_t)}.$$

En pratique : $s(q,d) = \sum_{t \in q \sqcap d} \log \frac{a+0.5}{c+O.5} \frac{d+0.5}{b+0.5}$ avec $a =$nb appartition terme dans doc pertinent, b nb apparition terme dans document non pertinent, c nb de **non** apparitition dans document pertinent et d nb **non** apparition terme dans document non pertinenet

	Doc Pertinent	Non Pertinent
$t \in d$	a	b
$t \notin d$	c	d

— On estime les proba par max vraissemblance qui donne juste la fréquence des termes
— On peut intégré un prior sur $P(d)$ mais je sais honnetement pas à quelle étape : longueur du doc, longueur moyenne des mots, date, nombre de lien, pagerank

7.1 BM25/Okapi

Si $t \notin d \Leftrightarrow TF_t = 0$ puis avec une modelisation de poisson sur la valeur de ce terme, on retombe sur la formule du coef BM25. Woah qu'est ce que c'est que ce trucs c'est le futur à estimer les param de la poisson

BM25 est un modèle de sac de mots qui ordonne les documents en fonction de la fréquence des termes qui apparaissent dans chaque document, indépendamment des relations pouvant exister entre ces termes ou de leurs proximités relatives au sein du document. Pour une requête Q contenant les mots $q_1, \ldots, q_n$, le score BM25 d'un document D est $s(D,Q) = \sum_{i=1}^{n} IDF(q_i) \frac{f(q_i,D)(k_1+1)}{f(q_i,D)+k_1(1-b+b\frac{|D|}{avgdl})}$ avec $f(q_i, D) = tf_i$ fréquence du terme q_i dans le document D, $|D|$ Longueur du doc D (nombre de mot), $avgdl$ longueur moyenne des documents de la collection, $k_1 = 1.2, b = 0.75$, $IDF(q_i) = \log \frac{N-n(q_i)+0.5}{n(q_i)+0.5}$ avec $n(q_i)$ le nombre de document contenant q_i

8 Modèle de langue

Basé sur l'idée que pour faire une recherche on imagine les mots que le documents pertinent vas contenir. Quel est la proba que le document soit généré par le même modèle de langue que le document.

$$p(t_1, \ldots, t_n) = \sum_t tf(t) \log P(t|\theta_{Md}) = \sum_t tf(t) \log p_t.$$

Par max vraissemblance + langragien $p_t = \frac{tf_(t)}{\sum_t tf(t)}$. C'est très très flou dans le diapo.

Dans le cas où un mot de la requête n'apparrait pas dans le document, score = 0 $\rightarrow$ Lissage des probas = modèle de mélange multinomial entre la distribution des termes dans le document et la distribution des termes dans la collection = Dirichlet ou Jelinek-Mercer

9 Reformulation de requete

9.1 Revelance feedback

Recalculer les poids des document en fonction du feedback des users et recalculer des nouveaux score. Modèle de Rocchio pour les modèles vectoriel

— Par le retour de l'utilisateur, deux ensembles de vecteur de document : les documents pertinent VS les non pertinents
— Vecteur moyen des documents pertinent $\rightarrow$ Correction de notre vecteur de question

$$q_{new} = aq_{old} + b * d^+ - cd^-.$$

avec d^+ le vecteur moyen des documents pertinents, same pour d^-. Puis binariser q avec un seuil
Limite :

— Fiabilité des users sur les retours positif négatif
— Comment ils évalue la pertinence
— Mais on garde un effet de masse qui moyenne les erreurs

9.2 Pseudo revelance feedback

Sugestion d'une nouvelle requete en se basant sur les k premier document. Pour la trouver méthode de clustering, similarité des termes, ...

Limite :

— Couteux
— Query drift : si les top documents ne sont pas pertinents, la requete reformulée ne reflètera jamais le besoin de l'utilisateur (exemple : Apple et apple : on vas biaiser la requete vers un seul des deux sens)

10 Métrique

10.1 Métrique orientées rappel/précision

Précision et rappel

— Recall : Pourcentage de documents pertinents renvoyés parmi tous ceux qui sont pertinents. Utilisé quand les faux négatifs sont couteux (exemple : le médical, documentaliste).

$$\frac{\|R \cap P\|}{\|P\|}.$$

Avec R l'ensemble des documents renvoyés et P les documents pertinents.

— Précision : Pourcentable de documents pertinents renvoyés parmis ceux renvoyés. Utilisé quand les faux positif sont couteux (exemple : la RI : moteur de recherche).

$$\frac{\|R \cap P\|}{\|R\|}.$$

Avec R l'ensemble des documents renvoyés et P les documents pertinents.

— C'est un compromi, augmenter l'un fait baisser l'autre

— F-mesure

$$F_\beta = (1+\beta^2)\frac{P * R}{\beta^2(P+R)}.$$

— Tracer une courbe précision recall :

1. faire un tableau avec pour colonne : rang, (probabilité), y true, recall, precision.
2. Calculer le recal et la précision pour chaque ligne en rajoutant les résultats d'avant
3. Puis pour tracer la courbe : chaque ligne = un point du graph

— Précision interpolée : super visuel, pour tracer la courbe, on fait varier le recall, pour chaque point on regarde la précision max à droite de ce point. Soit r un point de recall, $P_{interp}(r) = \max_{r' \geq r} P(r')$

— Précision moyenne : moyenne arithmétique de la précision interpolé
OU $\approx$ AveP $= \frac{\sum_{k=1}^{n} P(k) \times 1_{doc_k \text{ is revelant}}}{\text{total number of relevant documents}}$

— **Moyenne des précision moyenne (MAP)** = moyenne des précision moyenne pour chaque requette

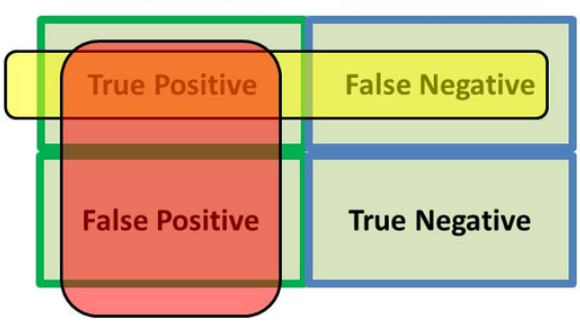

Figure 1 – En rouge la précision, en jaune le rappel

10.2 Métrique orientées rang

— **Moyenne des rangs inverse (MRR)** : moyenne de l'inverse du rang du premier document pertinent renvoyé sur l'ensemble des requêtes $\frac{1}{|Q|}\sum_{q \in Q} \frac{1}{rank_i}$ where rank$_i$ refers to the rank position of the first relevant document for the i-th query.

— **Discounted cumulative gain (DCG)** : Somme pondéré par $\frac{1}{\log_2(rang)}$ du score de pertinence des documents. $DCG_p = score_1 + \sum_{i=1}^{p} \frac{score_i}{\log_2 i}$ pour une requète.

— Normlized DCG : pour pouvoir faire une moyenne sur l'ensemble des requète il faut normaliser le DCG en utilisant le IDCG (liste idéale de résultat) $nDCG_p = \frac{DCG_p}{IDCG_p}$

11 Apprentissage de model

— Distillation : Faire apprendre un premier modèle moins gourmand en ressource, qui fera beaucoup de faux négatif, puis entrainer un autre modèle par dessus ces prédictions $\rightarrow$ plus robuste au bruit car il est pas directement exposé au faux negatif. Très efficace

11.1 Modèle dense / de représentation

— Représenter la query et l'input dans un espace latent (le même ou non) de même taille puis score de similarité (cos ou autre)
— **Gros overfit** car augmenter le score $\Leftrightarrow$ augmenter la norme et espace assez grand pour overfit chaque doc
— **forward lent**
— Technique de clusterisation :
  — Pour eviter l'overfit pendant l'apprentisage batch uniquement du même cluster
  — Pour faire moins de produit scalaire car couteux, produit scalaire avec représentatn des clusters

11.2 Modèle d'intération

— Intération faible : matrice pour faire la similarité into NN score (marchait pas beaucoup)
— Intéraction forte : concaténation de la query et du doc into NN score (cross encoder)
— Cross Encoder : Transformer + input concat query doc $\rightarrow$ Classifier linéaire sur la sortie en grande dimension $\rightarrow$ score ; gourmand mais performant
— Mécanisme en deux temps : gros tri avec un pré-traitement, et ordonancement avec le modèle d'interaction forte

11.3 Modèle sparce

— Rien compris
— Doc2query : Doc $\rightarrow$ Question sur le doc $\rightarrow$ concaténation de la question et du doc $\rightarrow$ amélioration des performances de recherche !
— CCL : Bon résultat, mieux que BM25, robuste ; un peu lourd à apprendre ; pour arriver à l'état de l'art il faut quand même ajouter un cross encoder en 2ème étape

12 Page Rank

— Analyse les liens entre les pages, graph orienté de page. une page a un PageRank d'autant plus important qu'est grande la somme des PageRanks des pages qui pointent vers elle
— On veut trouver la distribution stationnaire s du graph (comme dans markov chain) représentant l'importance de chaque page.
— A matrice d'ajascence avec $a_{ij} = 1$ si lien de i ver j, $d_i = \sum_j a_{ij}$ nombre de lien entrant pour une page, P matrice de transition avec $p_{ij} = \frac{a_{ij}}{d_i}$ proba de transition de j à i
— $s_j = d \sum_i p_{ij} s_i + (1-d) a_j, d$ facteur d'armotissement (≈ 0.8)
— $s = dsP + (1-d)a = s(dP + (1-d)\mathbf{1}) = s\hat{M}$
— sur wikipedia c'est toujours $s = dsP + (1-d) * \frac{1}{N}$, N nombre de doc.
— Version itérative :
  1. Initialisation de $S = \frac{1}{N}\mathbf{1}$ proba uniforme
  2. Calculer $\hat{M} = dP + \frac{1-d}{N}\mathbf{1}$
  3. $S_{n+1} = S_n * \hat{M}$ ou $S_{n+1} = dSP + \frac{1-d}{N}$