Correspondances compatibles avec les fichiers inverses pour la recherche d'information.

Philippe Mulhem, Jean-Pierre Chevallet

Correspondances compatibles avec les fichiers inverses pour la recherche d'information.

Philippe Mulhem, Jean-Pierre Chevallet

Université de Grenoble Alpes / CNRS, LIG UMR 5217, Grenoble, F-38041, France
{Philippe.Mulhem, Jean-Pierre.Chevallet}@imag.fr

RÉSUMÉ. *Cet article fait un retour sur l'un des éléments majeurs d'un système de recherche d'information : la correspondance basée sur des fichiers inverses car le passage d'une formule théorique à une implantation compatible avec des fichiers inverse est rarement explicitée dans les publications. Nous proposons ici de définir plus formellement l'expression d'une formule de correspondance compatible avec des fichiers inverses. Nous proposons deux niveaux de compatibilité. Nous étudions les modèles les plus classiques en vérifiant leur compatibilité avec les fichiers inverses. Nous explorons la traduction d'une correspondance de Jensen-Shannon, initialement non-compatible avec les fichiers inverses, vers deux formules compatibles avec les fichiers inverses à chaque niveau. Une expérimentation simple, sur un corpus d'images, montre que la classique Divergence de Kullback-Leibler obtient des résultats moins bons que la Divergence de Jensen-Shannon compatible avec des fichiers inverses.*

ABSTRACT. *This article focuses on one of the major elements of an information retrieval system: the matching function based on inverted files. In existing papers, the transition from a theoretical formula for implantation files compatible with the reverse is rarely explained. We define here formally what makes a matching function in IR compatible with inverted files. We define two levels of compatibility. We study classical matching functions and assess their inverted files compatibilities. We explore then the rewriting of the Jensen-Shannon divergence, initially not compatible with inverted files, into two inverted file compatible expressions. An experiment on an image corpus shows that the classical Kullback-Leibler divergence underperforms the Jensen-Shannon divergence compatible with inverted files.*

MOTS-CLÉS : *Fichier inverse, fonction de correspondance*

KEYWORDS: *Inverted file, matching function.*

1. Introduction

Cet article retourne sur l'un des éléments majeurs d'un système de recherche d'information : la correspondance basée sur des *fichiers inverses*. Depuis les débuts du domaine de la recherche d'information, la structure de fichier inverse a été reconnue comme un élément clé, capable de permettre à un système de fournir des réponses très rapidement même sur des grands corpus. Sans que cet élément ne soit remis en cause, le passage d'une formule théorique de correspondance à une expression compatible avec des fichiers inverses n'est que rarement expliquée : elle relève davantage d'un art que d'une méthode et est relativement difficile à cerner. Ainsi, nous proposons dans cet article, une méthode pour trouver un passage d'une formulation théorique de la correspondance vers l'implantation pratique à base de fichiers inverses.

Le plan de cet article est alors le suivant : un rappel sur les fichiers inverses est effectué, puis nous définissons deux compatibilités avec des fichiers inverses pour les fonctions de correspondances, permettant de proposer deux manières d'utiliser les fichiers inverses plus ou moins efficacement selon la formule théorique de correspondance. Nous décrivons ensuite des modèles classiques et leur expression sous forme compatibles avec des fichiers inverses. Une dernière étape explique comment une fonction de correspondance compatible peut être établie pour la négative de la divergence de Jensen-Shannon ; c'est à notre connaissance la première fois qu'une telle démarche est suivie pour cette fonction de correspondance. Nous terminons par montrer des résultats expérimentaux sur un corpus d'images avant de conclure.

2. Fichiers inverses

La notion de fichiers inverses en recherche d'informations a été décrite pour le système SMART en 1983 dans un article re-publié à la fin de années 90 par Salton et par Harman (Salton *et al.,* 1997, Harman *et al.,* 1992), où l'on peut lire en détails la création et l'utilisation de ces fichiers. Une telle structure tient compte du fait qu'une requête n'utilise qu'un nombre réduit de termes du vocabulaire : c'est la justification clé d'un fichier inverse.

L'indexation d'un corpus consiste à construire pour chaque document, une liste de couple : terme et poids du terme. Cette structure appelée *fichier direct*, nécessite le parcours exhaustif de tous les index des documents pour sélectionner le plus pertinents. Un fichier inverse permute l'accès du fichier direct : il ne se fait plus à partir des documents, mais à partir des termes. Un fichier inverse est alors pour chaque terme, une listes de couples : identifiant de document et poids du terme dans ce document. De manière plus formelle, nous avons :

$$fich\_inverse : T \longmapsto 2^{C_{id} \times \mathbb{R}} \qquad fich\_inverse(t) = \{< d, w >\}$$

avec T le vocabulaire, C_{id} les identifiants des documents du corpus et $\mathbb{R}$ l'ensemble des valeurs réelles. Dans le but de réduire la taille du fichier, un couple apparait uniquement quand le poids w du terme t dans le document d n'est pas une constante,

c'est-à-dire quand ce poids dépend réellement du document. Par exemple, avec une pondération de type tf.idf, seuls les termes apparaissant dans le document (i.e. $tf \neq 0$) donc avec un poids non nul sont donc présents dans le fichier inverse.

Avec une telle structure le traitement d'une requête limite l'accès aux seules listes de documents associés aux termes de la requête, contrairement au fichier direct où les listes de tous les documents doivent être lues séquentiellement. L'efficacité de cette structure est alors d'autant plus grande que le ratio entre la taille du vocabulaire et la taille des requêtes est important. Des informations supplémentaires peuvent être stockées dans le fichier inverse, comme la position des mots dans les documents, afin de prendre en compte la contiguïté des termes. Ces extensions sortent du cadre du travail décrit ici.

3. Fonctions de correspondances sur fichiers inverses

Nous venons donc de voir que si le nombre de documents est grand par rapport au vocabulaire, et si la taille du vocabulaire est grande par rapport à une requête (i.e. $|C| \gg |T| \gg |q|$), l'utilisation d'un fichier inverse devrait toujours être plus efficace qu'un fichier direct. La question que l'on pose est alors la suivante : dans quelle mesure un calcul de correspondance est-il compatible avec l'utilisation d'un fichier inverse ?

Lorsque l'on code la fonction de correspondance d'un SRI, la fonction théorique du calcul de la pertinence est toujours formulée indépendamment de toute implantation : pour une requête donnée, on fournit pour chaque document le moyen de calculer sa valeur de pertinence. Cette valeur de pertinence est calculée par composition (notée $\bigoplus$) sur tous les termes de T, d'une fonction élémentaire f_{elem} sur un seul terme t :

$$f(Q, D) = \bigoplus_{t \in T} f_{elem}(t, D) \qquad [1]$$

Le concepteur de la formule de correspondance se débrouille généralement pour que la composition soit une somme (le cas qui nous importe dans cet article), pour réduire les approximations des calculs. On note également que c'est plus l'ordre de pertinence des documents qui est important, plus que la valeur en elle même de la fonction de correspondance. C'est ainsi que la forme générale d'une formule de correspondance est :

$$f(Q, D) \stackrel{rang}{=} \sum_{t \in T} f_{elem}(t, D) \qquad [2]$$

Cette formulation, que nous appellerons la classe 0, avec $t \in T$, ne permet pas une utilisation optimale du fichier inverse, car il est nécessaire pour chaque requête de parcourir tout le vocabulaire. Une meilleure formulation consiste à trouver un calcul qui se limite aux termes de la requête : $t \in Q$. Nous l'appellerons la classe 1 de

compatibilité. De même, un nombre plus faible de calculs peut être obtenu si nous nous limitons aux termes de la requête qui sont aussi présents dans les documents : $t \in Q \cap D$. Nous nommerons cette compatibilité, la classe 2. Nous allons maintenant examiner plus en détails ces deux classes de compatibilité des formules avec un fichier inverse.

3.1. *Compatibilité de fichier inverse de classe 1 (appartenance)*

Une fonction de correspondance théorique $f(Q, D)$ entre une requête Q et un document D est *compatible avec un fichier inverse* si et seulement si f est exprimable sous la forme ci-dessous :

$$f(Q, D) \stackrel{rang}{=} K_D + \sum_{t \in Q} f_{elem}(t, D) \qquad [3]$$

La notation $\stackrel{rang}{=}$ indique que le tri des résultats de cette fonction sur les documents D du corpus par la fonction initiale f ne dépend que d'une fonction élémentaire qui se limite aux termes qui occurrent dans (ou **appartiennent à**) la requête. Un élément important de cette formule est la constante K_D : c'est une valeur qui dépend de données globales du document D et éventuellement de paramètres liés à la correspondance (paramètre de lissage par exemple). Dans tous les cas cette constante peut être calculée hors ligne avant une calcul de correspondance pour une requête donnée.

Cette description permet donc bien d'utiliser la structure de fichier inverse décrite dans la section précédente, car il est suffisant de ne passer en revue que les termes de la requête pour obtenir un résultat cohérent avec la formule théorique $f(Q, D)$. Par contre elle s'avère compliquée à gérer si la fonction $f_{elem}(t, D)$ est non constante pour un terme n'appartenant pas à D, impliquant son absence dans $fich\_inverse(t)$.

Un exemple de compatibilité de classe 1 se rencontre avec un modèle de langue avec similarité de requête (*Query Likelihood* dans (Zhai *et al.,* 2001)) basée sur l'hypothèse de distribution multinomiale. Quelque soit le lissage utilisé pour calculer la probabilité d'un terme dans un document, nous avons :

$$P(Q|D) \propto_Q \prod_{t \in Q} P(t|D)^{\#(t,Q)}$$

Avec $\#(t, Q)$ le nombre d'occurrences du terme t de la requête Q, et $\propto_Q$ indiquant que la proportionnalité ne dépend que de Q. Dans ce cas, quelque soit le lissage utilisé, on exprime la correspondance par :

$$logP(Q|D) \propto_Q \sum_{t \in Q} \#(t, Q) * logP(t|D)$$

Dans ce cas on a donc bien :

$$f(Q, D) \stackrel{rang}{=} K_D + \sum_{t \in Q} f_{elem}(t, D)$$

$$\text{avec} \quad f_{elem} = \#(t,Q) * logP(t|D) \quad \text{et} \quad K_D = 0$$

La difficulté avec une telle expression est que la probabilité $P(t|D)$ doit être évaluée même si le terme n'apparaît pas dans le document, et donc que le document n'apparaît pas dans la liste du fichier inverse correspondant au terme. Il en résulte un algorithme plus complexe lors du traitement de la requête :

– soit l'on passe en revue une première fois tout le fichier inverse pour déterminer la liste des documents de la réponse (ceux qui ont au moins une occurrence de l'un des termes de la requête), et dans une seconde étape on calcule les $P(t|D)$ pour tous les documents, même ceux qui ne sont pas dans la liste d'un terme ;

– soit le processus de correspondance procède en une passe, terme par terme, avec éventuellement des retours pour traiter les termes des documents non-rencontrés jusqu'alors.

Prenons l'exemple d'une requête avec 2 termes t_1 et t_2, passés en revue dans cet ordre dans le fichier inverse. Si un document D ne contient pas t_1, alors il n'est pas dans la liste du terme t_1 et le calcul de correspondance partiel de D pour t_1 n'est pas effectué. Si de plus ce document D contient t_2, alors il faut faire un retour pour calculer la partie partielle de la correspondance de D provenant de t_1. Cette démarche peut s'avérer compliquée pour un nombre quelconque de termes. Une variante de la version ci-dessus, stocke pour chaque document les termes de la requête qui ont permis de calculer la résultat partiel, et en fin de traitement les termes qui n'apparaissent pas dans le document sont intégrés.

La prise en compte d'une compatibilité de classe 1 n'est clairement pas la meilleure en termes de rapidité de traitement de requêtes et de complexité de l'algorithme. Celà nous amène à définir une notion de compatibilité de fichier inverse de class 2.

3.2. *Compatibilité de fichier inverse de classe 2 (intersection)*

Une fonction de correspondance théorique $f(D,Q)$ entre une requête Q et un document D est *compatible de classe 2 avec un fichier inverse* si et seulement si f est exprimable sous la forme ci-dessous :

$$f(Q,D) \stackrel{rang}{=} K_D + \sum_{t \in Q \cap D} f_{elem}(t,D) \qquad [4]$$

Cette description utilise de manière plus directe la structure de fichier inverse décrite dans la section précédente, car elle ne passe en revue que les termes de la requête qui sont également dans le document, et cela correspond exactement à ce que stocke le fichier inverse. On est donc dans le cas de la rpise en compte des termes qui sont à l'**intersection** du document et de la requête. La constante K_D est similaire à ce que nous avons décrit pour la compatibilité de classe 1.

Des exemples classiques de formules compatibles de classe 2 avec les fichiers inverses sont le modèle vectoriel (Salton *et al.,* 1975) et BM25 (Robertson *et al.,*

1994, Robertson *et al.,* 2009). Le cas de BM25 est simple, car la définition même de BM25 est :

$$BM25(Q,D) = \sum_{t \in Q} log(\frac{N - n_t + 0.5}{n_t + 0.5}).\frac{\#tf, D}{k_1.(1-b) + b.\frac{|D|}{avgdl}) + \#tf, D} \quad [5]$$

avec $avgdl$ la taille moyenne des documents, k_1 et b des paramètres fixés, et n_t le nombre de documents du corpus contenant le terme t. Cette formule est seulement compatible de classe 1, mais il suffit de constater que, quand un terme t n'apparaît pas dans le document, la valeur $\#tf, D$ est nulle, et donc :

$$BM25(Q,D) = \sum_{t \in Q \cap D} log(\frac{N - n_t + 0.5}{n_t + 0.5}).\frac{\#tf, D}{k_1.(1-b) + b.\frac{|D|}{avgdl}) + \#tf, D} \quad [6]$$

Pour une correspondance dans un modèle vectoriel à base de cosinus entre documents et requêtes, la formule est :

$$cos(Q,D) = \sum_{t \in T} \frac{w_{t,Q} \times w_{t,D}}{||Q|| * ||D||} \quad [7]$$

Quand un terme n'apparaît pas dans la requête, et que par hypothèse son poids $w_{t,Q}$ est nul (ce qui est opportunément le cas avec des pondérations $tf.idf$), nous obtenons une formule compatible de classe 1 avec les fichiers inverses :

$$cos(Q,D) = \sum_{t \in Q} \frac{w_{t,Q} \times w_{t,D}}{||Q|| * ||D||} \quad [8]$$

Quand, en plus, un terme t de la requête n'apparaît pas dans le document , alors pour les même raisons que ci-dessus (avec une pondération $tf.idf$) nous obtenons une formule compatible de classe 2 :

$$cos(Q,D) = \sum_{t \in Q \cap D} \frac{w_{t,Q} \times w_{t,D}}{||Q|| * ||D||} \quad [9]$$

Dans les exemples cités ci-dessus, le passage à une compatibilité de classe 2 est évident, en particulier car les formules originales utilisent des pondérations proportionnelles au tf, ce qui permet d'avoir la constante $K_D = 0$. On pourrait, même si cela n'est pas étudié ici, se poser la question de ce qui se passerait si le modèle vectoriel reposait sur des pondérations non nulles pour les termes absents des documents : dans ce cas le passage à un expression compatible de niveau 2 serait plus compliqué. C'est typiquement le problème qui se pose pour les modèles de langues. Les auteurs de (Zhai *et al.,* 2001) ont présentés des expressions compatibles de classe 2, mais sans détailler leur démarche. C'est pourquoi, nous présentons dans cet article, une démarche permettant d'adapter une formule de similarité générale pour en faire une expression compatible de classe 1 ou 2 quand c'est possible.

4. Démarche d'étude de compatibilité de fonctions de correspondance

Nous avons déjà effleuré une démarche d'étude dans les exemples précédents, mais en nous limitant à de simples réécritures pour des valeurs spécifiques. Nous proposons maintenant la description d'une démarche à suivre pour tenter de vérifier les classes de compatibilité. Pour cela, nous allons nous baser sur une formule générale de somme de parties élémentaires par terme, $f_{elem}(t, D)$, sous forme d'une fonction de correspondance théorique $f(Q, D)$ entre une requête et un document :

$$f(Q, D) = \sum_{t \in T} f_{elem}(t, D) \qquad [10]$$

Nous utilisons trois méthodes de base pour transformer cette fonction tout en garantissant que l'ordre des résultats n'est pas modifié :

– **Elément neutre** : une modification de la formule en intégrant un élément neutre (typiquement en ajoutant et soustrayant la même chose) est un moyen de réécrire une formule de manière plus adaptée au problème de compatibilité. Cet élément dépend éventuellement du document et de la requête ;

– **Extension** : dans ce cas nous étendons la formule avec une même constante pour tous les documents. Une telle extension, est *conservative* : elle ne modifie pas l'ordre des réponses basées sur la formule initiale ;

– **Approximation** : si une partie de la formule possède une limite qui est fixée pour des valeurs correspondant aux plages de valeurs dans la formule, alors on peut dire que l'ordre de la formule modifiée sera similaire (mais non forcément égal) à l'ordre qui serait obtenu par la formule initiale. On note que dans ce cas nous n'aurons donc plus un égalité d'ordre $\overset{rang}{=}$ mais une approximation de l'égalité notée $\overset{rang}{\approx}$.

Comme habituellement avec l'écriture de démonstrations, savoir comment et quand utiliser ces méthodes relève, pour une grande part, de l'expérience. Nous tentons cependant d'expliciter ces étapes dans la suite, pour des correspondances sur des modèles de langue.

5. Application à des modèles de langues de RI

Nous commençons par nous intéresser à divergence de Kullback-Leibler dans les modèles de langues (qui généralise l'approche de probabilité de génération de requêtes, *Query Likelihood*), avant de passer à une autre fonction de similarité qui est la divergence de Jensen-Shannon, qui à notre connaissance n'a pas été étudiée en recherche d'information, peut-être justement pour sa non-adaptation aux fichiers inverses.

5.1. *Divergence de Kullback-Leibler*

Cette partie est fortement inspirée de (Zhai *et al.,* 2001), mais transposée à la négative de la divergence de Kullbak-Leibler. Le cas de calcul de divergence pose déjà des questions sur les compatibilité de classe 1, car pour la négative de Kullback-Leibler ce calcul se porte sur toutes les variables aléatoires, et non pas uniquement celles qui correspondent aux termes de la requête. La divergence négative de Kullback-Leibler est exprimée par :

$$nKL(Q||D) = -\sum_{t \in T} P(t|Q).\log \frac{P(t|Q)}{P(t|D)} \quad [11]$$

En supposant que les probabilités pour les documents sont lissées, et que la probabilité pour la requête est P_{LM}, on peut réécrire la formule en éliminant les termes non-présents dans la requête :

$$nKL(Q||D) = -\sum_{t \in Q} P(t|Q).\log \frac{P(t|Q)}{P(t|D)} \quad [12]$$

La formule [12] est compatible de classe 1 avec des fichiers inverses. Pour vérifier une compatibilité de classe 2, nous développons le log de l'expression ci-dessus :

$$nKL(Q||D) = \sum_{t \in Q} P(t|Q).\log P(t|D) - \sum_{t \in Q} P(t|Q).\log P(t|Q) \quad [13]$$

La seconde partie de la formule ci-dessus est une constante pour une requête donnée (en fait l'entropie de la requête), on peut donc l'éliminer en gardant l'ordre des réponses à une requête. Ensuite, sur cette première partie, on peut séparer, en reprenant les notations précédentes, la partie qui correspond aux termes présents dans le document des autres. Pour aider la suite de démonstration, nous séparons explicitement les probabilités pour les termes présents ("seen terms") dans un document $P_s(t|D)$, des probabilités pour les termes non-présents ("unseen terms") dans les documents $P_u(t|D)$, quand cela est nécessaire. Ces notations sont tirées de (Zhai *et al.,* 2001).

$$nKL(Q||D) \stackrel{rang}{=} \sum_{t \in Q \cap D} P(t|Q).\log P_s(t|D) + \sum_{t \in Q \setminus D} P(t|Q).\log P_u(t|D) \quad [14]$$

Pour traiter la seconde partie qui dépend des termes présents dans la requête mais pas dans le document, nous ajoutons l'élément neutre suivant :

$$\sum_{t \in Q \cap D} P(t|Q).\log P_u(t|D) - \sum_{t \in Q \cap D} P(t|Q).\log P_u(t|D)$$

Ce qui donne après simplification des écritures :

$$nKL(Q||D) \stackrel{rang}{=} \sum_{t \in Q \cap D} P(t|Q).\log \frac{P_s(t|D)}{P_u(t|D)} + \sum_{t \in Q} P(t|Q).\log P_u(t|D) \quad [15]$$

L'étape suivante s'intéresse à la seconde partie de l'expression ci-dessus. Elle est en fait similaire à ce que nous avons fait avec les probabilités de génération des requêtes, en détaillant les probabilités $P_u(t|D)$.

Dans le cas de lissage de Jelinek-Mercer, l'expression ci-dessus devient :

$$nKL(Q||D) \overset{rang}{=} \sum_{t \in Q \cap D} P(t|Q).\log \frac{P_s(t|D)}{P_u(t|D)} + \sum_{t \in Q} P(t|Q).\log \lambda.P_{ML}(t|C) \quad [16]$$

avec P_{ML} une probabilité estimée par maximum de vraisemblance. Comme ce second élément de cette expression dépend uniquement de la requête et du corpus, il n'influe pas sur l'ordre des réponses, nous obtenons une expression compatible de classe 2 :

$$nKL(Q||D) \overset{rang}{=} \sum_{t \in Q \cap D} P(t|Q).\log \frac{P_s(t|D)}{P_u(t|D)} \quad [17]$$

Dans le cas d'un lissage de Dirichlet, nous reprenons l'expression [15], en explicitant P_u dans la seconde partie :

$$nKL(Q||D) \overset{rang}{=} \sum_{t \in Q \cap D} P(t|Q).\log \frac{P_s(t|D)}{P_u(t|D)} + \sum_{t \in Q} P(t|Q).\log(\mu.P_{ML}(t|C)) - \sum_{t \in Q} P(t|Q).\log(|D| + \mu) \quad [18]$$

La deuxième partie est constante pour une requête, on la retire :

$$nKL(Q||D) \overset{rang}{=} \sum_{t \in Q \cap D} P(t|Q).\log \frac{P_s(t|D)}{P_u(t|D)} - \sum_{t \in Q} P(t|Q).\log(|D| + \mu) \quad [19]$$

Une dernière simplification d'écriture nous donne finalement :

$$nKL(Q||D) \overset{rang}{=} \sum_{t \in Q \cap D} P(t|Q).\log \frac{P_s(t|D)}{P_u(t|D)} - \log(|D| + \mu) \quad [20]$$

Ce qui est bien une expression compatible de classe 2 :

$$nKL(Q||D) \overset{rang}{=} \sum_{t \in Q \cap D} P(t|Q).\log \frac{P_s(t|D)}{P_u(t|D)} + K_D \quad [21]$$

avec $K_D = -\log(|D| + \mu)$.
Pour résumer :

– la compatibilité de classe 1 pour la négation de la Divergence de Kullback-Leibler a été montrée en utilisant simplement des réécritures dans l'équation [12] ;

– la compatibilité de classe 2, obtenue par l'équation [21], a nécessité l'utilisation d'un élément neutre.

Ces calculs ont été réalisés de manière relativement simple.

5.2. *Divergence de Jensen-Shannon*

La divergence de Jensen-Shannon a pour avantage d'être bornée et d'être symétrique. À notre connaissance, cette divergence n'a pas été proposée pour la recherche d'information, à notre avis en raison du manque d'implantation sur des fichiers inverses. C'est pour cela que nous nous y intéressons ici.

La divergence de Jensen Shannon, comme définie dans (Lin, 1991), évalue dans quelle mesure deux distributions de probabilité Q et D, représentée par $q_t = P(t|Q)$ et $d_t = P(t|D)$ sont différentes :

$$JSD_{\pi_Q}(Q, D) = \pi_Q.KL(Q \parallel \overline{QD}) + \pi_D.KL(D \parallel \overline{QD}) \quad [22]$$

avec

– Une relation de dépendance entre les deux constantes : $\pi_Q + \pi_D = 1$;

– La divergence de Kullback-Leibler $KL(. \parallel .)$ décrite plus haut ;

– La moyenne pondéré $\overline{QD}$ de D et Q, définie par :

$$\overline{qd}_t = \pi_Q.q_t + \pi_D.d_t \quad [23]$$

Nous estimons que cette divergence, en accordant un rôle symétrique à la requête et aux documents, est susceptible de fournir des résultats intéressants pour un système de recherche d'information.

5.3. *Compatibilité de classe 1*

En réécrivant la négation de la formule [22] en s'inspirant de ce qui a été fait pour le divergence de Kullback-Leibler, la formule qui définie la divergence négative de Jensen-Shannon est :

$$\begin{aligned} nJSD_{\pi_Q}(Q,D) = & -\pi_Q.\sum_{t\in T} q_t log \frac{q_t}{(\pi_D.d_t + \pi_Q.q_t)} \\ & -\pi_D.\sum_{t\in T} d_t log \frac{d_t}{(\pi_D.d_t + \pi_Q.q_t)} \\ = & -\pi_Q.\sum_{t\in T} q_t log\ q_t + \pi_Q.\sum_{t\in T} q_t log\ (\pi_D.d_t + \pi_Q.q_t) \\ & -\pi_D.\sum_{t\in T} d_t log\ d_t + \pi_D.\sum_{t\in T} d_t log\ (\pi_D.d_t + \pi_Q.q_t) \end{aligned} \quad [24]$$

La divergence de Jensen Shannon compare deux distributions en se basant sur leur moyenne pondérée. La nJSD nécessite donc que chaque $\pi_Q.q_t + \pi_D.p_t$ soit non nul. D'autre part, il a été montré (Österreicher *et al.,* 2003, Fuglede *et al.,* 2004) que pour $\pi_Q = 0.5$ la racine carrée de la divergence est une distance, et la JSD est bornée (Lin, 1991). Les poids π_Q and π_D définissent un a priori sur le fait que l'une ou l'autre distribution est choisie. Quand $\pi_Q = 0$ ou $\pi_D = 0$ la divergence de Jensen Shannon est égale à 0, et quand π_Q tend, vers 0, le comportement attendu de la JSD est proche de KLD. Une valeur π_Q plus grande (resp. plus petite) que 0.5 dénote que la requête (resp. le document) est la distribution la plus probable.

Nous éliminons la partie de [24] qui ne dépend que de la requête, ce qui donne :

$$\begin{aligned} nJSD_{\pi_Q}(Q,D) \stackrel{rang}{=} & \pi_Q.\sum_{t\in T} q_t log\ (\pi_D.d_t + \pi_Q.q_t) \\ & -\pi_D.\sum_{t\in T} d_t log\ d_t + \pi_D.\sum_{t\in T} d_t log\ (\pi_D.d_t + \pi_Q.q_t) \end{aligned} \quad [25]$$

Nous regroupons la première et la dernière partie de la formule ci-dessus :

$$\begin{aligned} nJSD_{\pi_Q}(Q,D) \stackrel{rang}{=} & \sum_{t\in T} (\pi_Q.q_t + \pi_D.d_t).log\ (\pi_D.d_t + \pi_Q.q_t) \\ & -\pi_D.\sum_{t\in T} d_t log\ d_t \end{aligned} \quad [26]$$

Nous faisons, comme nous l'avons fait pour le nKL, une séparation entre les termes qui sont dans la requête ($t \in Q$) et termes non-présents dans la requête ($t \notin Q$) :

$$nJSD_{\pi_Q}(Q,D) \overset{rang}{=} \sum_{t \in Q} (\pi_Q.q_t + \pi_D.d_t).log\ (\pi_D.d_t + \pi_Q.q_t) + \sum_{t \notin Q} (\pi_D.d_t).log\ (\pi_D.d_t) - \pi_D.\sum_{t \in T} d_t log\ d_t \quad [27]$$

Si nous supposons que la distribution de probabilité du document suit une loi multinomiale, nous avons alors $\sum_{t \in T} d_t = 1$, et donc $\pi_D.\sum_{t \in T} d_t.log\ \pi_D$ est une constante quel que soit le document. Nous faisons donc une **extension** de la formule ci-dessus en soustrayant cette valeur de la formule [27], ce qui donne :

$$nJSD_{\pi_Q}(Q,D) \overset{rang}{=} \sum_{t \in Q} (\pi_Q.q_t + \pi_D.d_t).log\ (\pi_D.d_t + \pi_Q.q_t) + \sum_{t \notin Q} (\pi_D.d_t).log\ (\pi_D.d_i) - \pi_D.(\sum_{t \in T} d_t log\ d_t + \sum_{t \in T} d_t.log\ \pi_D) \quad [28]$$

En simplifiant la dernière ligne dans [28], nous obtenons :

$$nJSD_{\pi_Q}(Q,D) \overset{rang}{=} \sum_{t \in Q} (\pi_Q.q_t + \pi_D.d_t).log\ (\pi_D.d_t + \pi_Q.q_t) + \sum_{t \notin Q} (\pi_D.d_t).log\ (\pi_D.d_t) - \pi_D.\sum_{t \in T} d_t.log(d_t.\pi_D) \quad [29]$$

Les deux dernières partie de l'équation [29] s'annulent pour les termes t qui ne sont pas dans la requête Q. Il en découle l'expression finale de la négation de la divergence de Jensen Shannon pour la compatibilité de classe 1, sans avoir besoin d'utiliser de constante K_D :

$$nJSD_{\pi_Q}(Q,D) \overset{rang}{=} \sum_{t \in Q} \Big((\pi_Q.q_t + \pi_D.d_t).log\ (\pi_D.d_t + \pi_Q.q_t) - (\pi_D.d_t).log\ (\pi_D.d_t) \Big) \quad [30]$$

Cette expression ressemble à l'expression de la divergence par l'entropie de (Lin, 1991), mais elle se limite aux termes présents dans la requête. L'équation [30] est donc compatible de classe 1 avec des fichiers inverses.

5.4. *Compatibilité de classe 2*

Nous allons, dans cette partie, utiliser une approximation pour pouvoir exprimer le JSD sous une forme compatible de classe 2 avec les fichiers inverses. Nous allons, à partir de l'expression [30], trouver un moyen de négliger la partie qui correspond aux termes qui n'apparaissent pas dans le documents mais qui ont une probabilité non nulle à cause d'un lissage. Pour simplifier la suite, nous utilisons les notations suivantes :

$$d'_t = \pi_D.d_t\,; \qquad q'_t = \pi_Q.q_t.$$

Supposons pour la suite la condition nécessaire : π_D and π_Q sont du même ordre de magnitude. Nous reformulons [30] :

$$nJSD_{\pi_Q}(Q,D) \stackrel{rang}{=} \sum_{t\in Q} q'_t.log\ (d'_t + q'_t) + d'_t.log\ (1 + \frac{q'_t}{d'_t}) \qquad [31]$$

En retranchant de [31] une constante pour une requête Q égale à : $\sum_{t\in Q} q'_t.log(q'_t)$. Nous obtenons :

$$nJSD_{\pi_Q}(Q,D) \stackrel{rang}{=} \sum_{t\in Q} q'_t.log\ (1 + \frac{d'_t}{q'_t}) + d'_t.log\ (1 + \frac{q'_t}{d'_t}) \qquad [32]$$

Comme précédemment, nous séparons les termes qui apparaissent à la fois dans le document et la requête (i.e. $t \in Q \cap D$), et ceux qui apparaissent dans la requête mais pas dans le document (i.e. $t \in Q \setminus D$) :

$$\begin{aligned} nJSD_{\pi_Q}(Q,D) \stackrel{rang}{=} & \sum_{t\in Q\cap D} q'_t.log\ (1 + \frac{d'_t}{q'_t}) + d'_t.log\ (1 + \frac{q'_t}{d'_t}) \\ & + \sum_{t\in Q\setminus D} q'_t.log\ (1 + \frac{d'_t}{q'_t}) + d'_t.log\ (1 + \frac{q'_t}{d'_t}) \end{aligned} \qquad [33]$$

Nous nous concentrons sur la deuxième ligne de la formule [33]. Pour un terme t absent de D, la valeur d'_t correspond à une valeur de lissage. Nous supposons que d'_t est beaucoup plus petit que q'_t, ce qui est le cas pour tout terme discriminant : $\frac{d'_t}{q'_t} << 1$. Nous utilisons dès lors le premier terme du développement de Taylor de $\ln(1+x)$ pour obtenir :

$$\begin{aligned} \sum_{t\in Q\setminus D} q'_t.\log\ (1 + \frac{d'_t}{q'_t}) &\approx \sum_{t\in Q\setminus D} q'_t.\frac{d'_t}{\ln(2).q'_t} \\ &\approx \sum_{t\in Q\setminus D} \frac{d'_t}{\ln(2)} \end{aligned} \qquad [34]$$

Cette somme est négligeable par rapport aux autres éléments de la formule, car ses valeurs sont liées aux probabilités lissées pour le document.

Considérons maintenant la dernière partie de la formule[33], nous avons :

$$\sum_{t \in Q \setminus D} d'_t.log\ (1 + \frac{q'_t}{d'_t}) \leq \sum_{t \in Q \setminus D} d'_t.log\ (1 + \frac{1}{d'_t})$$

$$= \sum_{t \in Q \setminus D} d'_t.(log\ (d'_t + 1) - log(d'_t)) \qquad [35]$$

Sachant que : $\lim_{\substack{x \to 0 \\ x > 0}} x.(log\ \ (x + 1) - log(x)) = 0$, nous déduisons que la valeur ci-dessus est négligeable par rapport aux autres parties de la formule globale.

Nous négligeons donc les parties [34] et [35] dans [33], ce qui nous donne l'approximation suivante [36] dans le cas d'un lissage de Jelinek-Mercer ou de Dirichlet :

$$nJSD_{\pi_Q}(Q, D) \overset{rang}{\approx} \sum_{t \in Q \cap D} \Big(\pi_Q.q_t.log\ (1 + \frac{\pi_D.d_t}{\pi_Q.q_t})$$

$$+ \pi_D.d_t.log\ (1 + \frac{\pi_Q.q_t}{\pi_D.d_t}) \Big) \qquad [36]$$

Cette dernière expression, et compatible de classe 2 avec les fichiers inverses et n'utilise pas de constante K_D. Elle a été est obtenue par deux approximations.

6. Expérimentations

Des tests effectués sur des documents textes sur la collection TREC-6 (Voorhees *et al.,* 2000) actuellement ne donnent pas de bons résultats : en terme MAP, les meilleurs résultats de JSD, comparés à un simple modèle utilisant KLD avec 0,3267, ont obtenu une MAP de 0,2327 (-51,02 %). Par contre les résultats obtenus sur des recherches d'images inspirée de (Pham *et al.,* 2011) sur la collection de bâtiments de Zürich appelée *Zubud* (Shao *et al.,* 2003) sont positivement significatifs. Cette collection est composée d'un corpus de 1005 images de 201 bâtiments, et d'un ensemble de test de 115 images requêtes. Des résultats reportés dans (Obdrzálek *et al.,* 2003) atteignent 100%, la problématique liée à cette collection pourrait donc être considérée comme résolue, cependant notre objectif ici est vérifier qu'utiliser la nJSD (sous sa forme compatible de classe 2) donne de bons résultats en terme de MAP comparé à l'utilisation habituelle de nKLD. Les caractéristiques visuelles extraites sont des SIFT couleurs sur un échantillonnage spatial dense, en utilisant le logiciel développé par Koen van De Sande (van de Sande *et al.,* 2010). Un regroupement (*clustering*) des caractéristiques est effectué par un algorithme de K-moyennes, définissant un vocabulaire de taille 2000 dimensions. Cette approche est habituellement appelée Sac de Mots Visuels (cf. (Csurka *et al.,* 2004)). Pour le modèle de langues considéré, nous avons testé des lissages de Dirichlet, avec μ de 300 à 4000 par pas de 100, et pour la nJSD des valeurs π_Q allant de 0,1 à 0,9 par pas de 0,1 . Le tableau 1 présente les meilleurs

résultats, pour nKLD avec $\mu = 1000$, et pour nJSD avec $\pi_Q = 0,6$ et $\mu = 2200$. On note que la nJSD donne de meilleurs résultats que nKLD. Pour les valeurs de MAP, le divergence de Jensen-Shannon donne de meilleurs résultats que le divergence de Kullback Leibler, avec +4,82 %, et cette différence est hautement statistiquement significative selon un test de Student par paires bilatéral, avec $p = 0,008 < 0,01$. Ces résultats sont confirmés également sur les valeurs de Rang Réciproque Moyen (RRM) ainsi que celles de précision à 10 documents.

Tableau 1. *MAP, Rang Réciproque Moyen et précision à 10 documents pour la collection Zubud*

divergence (paramètres)	MAP	RRM	P@10
nKLD ($\mu = 100$)	0,6998	0,9072	0,3826
nJSD ($\mu = 2200, \pi_Q = 0,6$)	0,7335 (+4,82%)	0,9319 (+2,72%)	0,3930 (+ 2,66%)

7. Conclusion

Nous avons proposé dans cet article un modèle de compatibilité avec un fichier inverse pour les formules de score pour la correspondance entre documents et requêtes. La compatibilité d'une formule de correspondance avec une structure de fichiers inverses, est fondamentale pour construire un Système de Recherche d'Information opérationnel performant. Nous avons défini deux classes de compatibilités, suivant que les termes de la fonction dépendent uniquement des termes présents dans la requête (compatibilité de classe 1) ou bien de l'intersection entre les termes de la requête et des documents (compatibilité de classe 2). Nous avons ensuite fait une retour sur la majorité des fonctions de score classiques (modèle vectoriel, BM25, modèles de langues). Nous avons ensuite défini une fonction de score basée sur la divergence négative de Jensen-Shannon, et explicité la manière d'approximer cette divergence en respectant une compatibilité de classe 2 avec les fichiers inverses. Nous avons enfin expérimenté sur un corpus d'images courant pour montrer l'intérêt potentiel de la divergence de l'expression de Jensen-Shannon compatible avec des fichiers inverses. Les travaux futurs liés aux résultats obtenus vont raffiner la fonction de score basée sur le divergence de Jensen-Shannon, afin d'estimer plus précisément ses avantages et inconvénients pour la recherche d'information de textes et de documents multimédia. De plus, comme nous l'avons signalé sans le détailler, les résultats de nJSD sont moins bons sur des collections de textes, et nous devons étudier pourquoi cela se produit.

Remerciements

La auteurs rémercient la région Rhône Alpes, qui a partiellement subventionné ce travail dans la cadre du projet ExploraDoc.

8. Bibliographie

Csurka G., Dance C. R., Fan L., Willamowski J., Bray C., « Visual categorization with bags of keypoints », *In Workshop on Statistical Learning in Computer Vision, ECCV*, p. 1-22, 2004.

Fuglede B., Topsoe F., « Jensen-Shannon divergence and Hilbert space embedding », *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, p. 31, june-2 july, 2004.

Harman D., Baeza-Yates R., Fox E., Lee W., « Information retrieval », Prentice-Hall, Inc., Upper Saddle River, NJ, USA, chapter Inverted files, p. 28-43, 1992.

Lin J., « Divergence measures based on the Shannon entropy », *Information Theory, IEEE Transactions on*, vol. 37, n° 1, p. 145 -151, jan, 1991.

Obdrzálek S., Matas J., « Image Retrieval Using Local Compact DCT-Based Representation. », *in* , B. Michaelis, , G. Krell (eds), *DAGM-Symposium*, vol. 2781 of *Lecture Notes in Computer Science*, Springer, p. 490-497, 2003.

Österreicher F., Igor V., « A new class of metric divergences on probability spaces and its statistical applications », *Ann. Inst. Statist. Math.*, vol. 55, n° 3, p. 639-653, 2003.

Pham T.-T., Mulhem P., Maisonnasse L., Gaussier E., Lim J.-H., « Visual Graph Modeling for Scene Recognition and Mobile Robot Localization », *Multimedia Tools and Applications*p. 1-23, Jan, 2011. Springer Science.

Robertson S. E., Walker S., Jones S., Hancock-Beaulieu M., Gatford M., « Okapi at TREC-3 », *TREC*, p. 0-, 1994.

Robertson S., Zaragoza H., « The Probabilistic Relevance Framework : BM25 and Beyond », *Found. Trends Inf. Retr.*, vol. 3, n° 4, p. 333-389, April, 2009.

Salton G., McGill M. J., « Readings in information retrieval », Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, chapter The SMART and SIRE experimental retrieval systems, p. 381-399, 1997.

Salton G., Wong A., Yang C. S., « A vector space model for automatic indexing », *Commun. ACM*, vol. 18, n° 11, p. 613-620, November, 1975.

Shao H., Svoboda T., Gool L. V., ZuBuD — Zürich buildings database for image based recognition, Technical Report n° 260, Computer Vision Laboratory, Swiss Federal Institute of Technology, March, 2003.

van de Sande K. E. A., Gevers T., Snoek C. G. M., « Evaluating Color Descriptors for Object and Scene Recognition », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, n° 9, p. 1582-1596, 2010.

Voorhees E. M., Harman D., « Overview of the sixth text REtrieval conference (TREC-6) », *Inf. Process. Manage.*, vol. 36, n° 1, p. 3-35, January, 2000.

Zhai C., Lafferty J., « A study of smoothing methods for language models applied to Ad Hoc information retrieval », *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, ACM, New York, NY, USA, p. 334-342, 2001.