Le cosinus de Salton : un classique (méconnu) des moteurs de recherche

Retour au sommaire de la lettre

Domaine :	Recherche	**Référencement**
Niveau :	Pour tous	**Avancé**

A la genèse des moteurs de recherche, on trouve plusieurs méthodes de calcul de la pertinence d'une page par rapport à une requête donnée. Et parmi ces méthodes, la plus connue est certainement celle du cosinus de Salton, expliquée et illustrée dans cet article. Il est difficile de dire si Google et Bing l'utilisent encore, mais la compréhension de cette technique, certes âgée de 40 ans, reste essentielle dans la compréhension du fonctionnement des moteurs de recherche modernes...

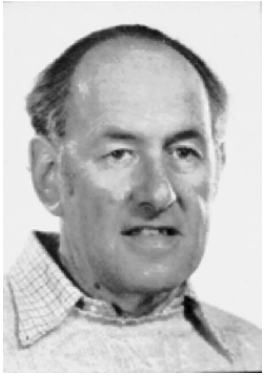
Nota : certaines URL correspondant aux sources citées dans cet article se trouvent à la fin.

Les moteurs de recherche construits depuis les années 60 ont implémenté de nombreuses techniques inventées par des linguistes, et en particulier des outils de statistique linguistique. Parmi ces outils, le plus connu est sans conteste le Cosinus de Salton, utilisé en particulier dans les années 90 par feu le moteur AltaVista.

On ne sait pas vraiment si Google utilise également cet outil mathématique dans son algorithme : les techniques ont beaucoup évolué depuis, et il est probable que le moteur de recherche de Google utilise des méthodes plus « modernes » et plus sophistiquées pour classer ses pages. Mais les principes à l'œuvre dans le Cosinus de Salton sont intéressants à connaître pour tous ceux qui s'intéressent aux moteurs de recherche et leurs rouages internes.

Gerard (Gerry) Salton

Gerard Salton était un chercheur informatique de l'université de Cornell. D'origine allemande (son vrai nom est : Gerhard Anton Sahlmann, qu'il a américanisé en Gerard Salton), il a inventé le concept de « modèle vectoriel », le poids « TF*IDF » et le fameux Cosinus de Salton. Il est également à l'origine de la première implémentation pratique de ces outils dans un moteur, le fameux système « SMART » développé à l'université de Harvard. Il est décédé en 1995.
Ses travaux ont inspiré de très nombreuses recherches ultérieures jusqu'à aujourd'hui.



A la recherche d'un outil de calcul simple pour classer des documents.

Les premiers moteurs de recherche ne savaient que retourner la liste des documents qui contenaient un certain mot clé. On pouvait perfectionner la recherche en cherchant plusieurs mots clés à la fois en utilisant les opérateurs booléens « OU » et « ET » (« assurance ET automobile », ou « assurance OU automobile »). Cette technique devient rapidement inutilisable dès que l'on cherche au sein d'un grand nombre de documents. Evidemment, la meilleure solution est de ne pas retourner une liste en « vrac » mais de pouvoir classer les documents en faisant apparaître en premier les « meilleures » réponses.

Mais comment évaluer les réponses ? La première idée fut d'utiliser la fréquence d'apparition des termes dans le document (ce que l'on appelle en jargon SEO la « densité

de mots clés »). Mais Karen Spärck Jones, une chercheuse anglaise, a introduit en 1972 la notion d'IDF (*Inverse Document Frequency*). La « fréquence dans les documents » mesure en fait le nombre de documents qui contiennent un terme donné, rapporté à l'ensemble des documents analysés. On utilise dans la pratique plutôt l'inverse de ce nombre (IDF = 1/DF), car il sert à pondérer les fréquences d'apparition trouvées dans un texte, en tenant compte du fait que la fréquence d'apparition d'un terme dans un texte dépend aussi de la fréquence d'utilisation de ce terme dans l'ensemble des textes.

Gérard Salton et son équipe ont perfectionné le concept introduit par Karen Spärck Jones en créant la formule TF*IDF, qui donne pour un terme donné trouvé dans un document donné un « poids » qui révèle si le document est particulièrement intéressant à renvoyer pour une requête sur ce terme.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

Au fil des années, la formule de poids TF*IDF a été perfectionnée, de nombreuses variantes ont été inventées et testées. L'une de celles qui a fourni les meilleurs résultats dans un moteur de recherche est connue sous le nom "Okapi BM25".

$$BM25 = \sum_{i=1}^{W} \frac{TF(i)(1+k)}{TF(i)+k\left(1-b+b\frac{DL}{avgDL}\right)} IDF(i)$$

$$IDF(i) = \frac{\log\left(\frac{N-n+1}{n}\right)}{\log(N)}$$

La formule du poids BM25

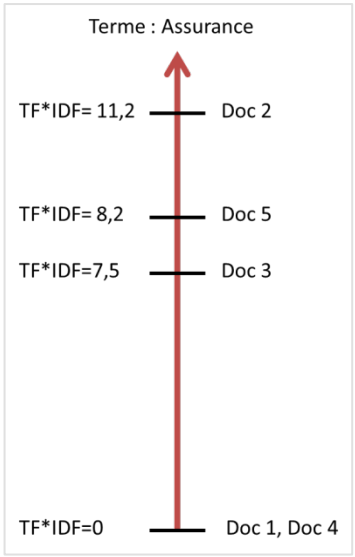
Classer les documents en fonction de leur poids sur un terme

Une fois que l'on dispose d'un poids pour un terme dans un document, on dispose d'un moyen simple pour classer les documents renvoyés par une requête sur un terme donné. En effet, il suffit de calculer le poids des termes lors de l'indexation, et de stocker ces poids en regard des termes dans l'index inversé.

Doc. Id	Term	Term count	TF	IDF	Term weight = TF*IDF
D1	user	83	1.92427	5.78996	11.141500423981
D1	information	78	1.89762	5.78996	10.987185277787
D1	personalization	49	1.69897	5.78996	9.8369686566546
D1	users	42	1.63346	5.78996	9.4577172982229
D2	content	33	1.53147	5.78996	8.8672019322438
D1	services	26	1.43136	5.78996	8.2875391845462
D1	quot	26	1.43136	5.78996	8.2875391845462
D1	service	25	1.41497	5.78996	8.1926393276321
D1	used	20	1.32221	5.78996	7.6555970537011
D1	site	17	1.25527	5.78996	7.2679778081705
D1	uk	17	1.25527	5.78996	7.2679778081705
D1	needs	16	1.23044	5.78996	7.1242502471038
D1	provide	16	1.230448	5.78996	7.1242502471038
D1	profile	16	1.230448	5.78996	7.1242502471038

*Exemple de calcul de poids tf*idf pour les termes dans un document*

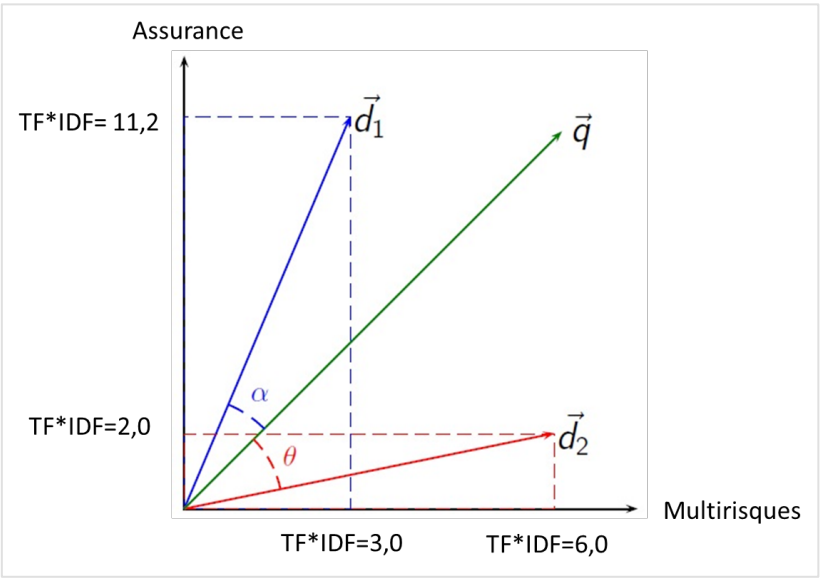
Lors de la requête, on peut ensuite classer dynamiquement les pages en fonction des poids relevés dans les documents pour un terme donné.



Disposer des poids du terme « assurance » dans un groupe de 5 documents permet de classer facilement ces documents sur la requête « assurance » dans l'ordre des poids décroissants. Deux documents ne contiennent pas le terme « assurance » et se voient donc attribuer un poids de zéro.

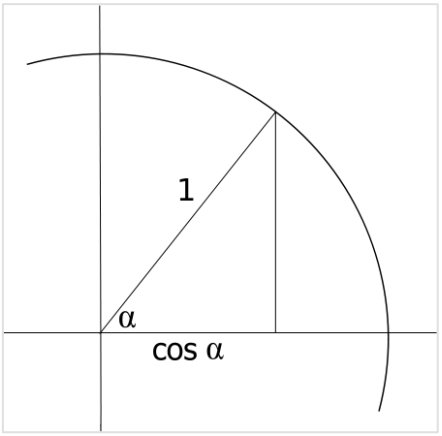
Challenge suivant : classer les documents en fonction de leur poids sur deux termes différents

Dans la pratique, les utilisateurs doivent pouvoir effectuer des requêtes contenant plusieurs termes. Par exemple, peut-on utiliser les poids pour classer les documents en fonction de leurs poids sur deux termes, comme dans « assurance multirisques » ?
Si l'on veut représenter la situation, on va devoir utiliser une représentation sur deux axes (un pour les poids sur le terme « assurance », un pour les poids sur le terme « multirisques ». Chaque document est donc caractérisé par des coordonnées sur deux axes. Ce type de représentation définit donc un « vecteur » pour chaque document, qui part de l'origine des deux axes et dont l'extrémité correspond au point dont les coordonnées sont définies par les poids respectifs sur chaque axe.



Cette représentation montre comment on peut créer un critère de classement opérationnel : le document qui présente les meilleurs TF*IDF sur les deux termes à la fois (le document d1) est celui qui a la distance angulaire la plus faible par rapport à la requête q.

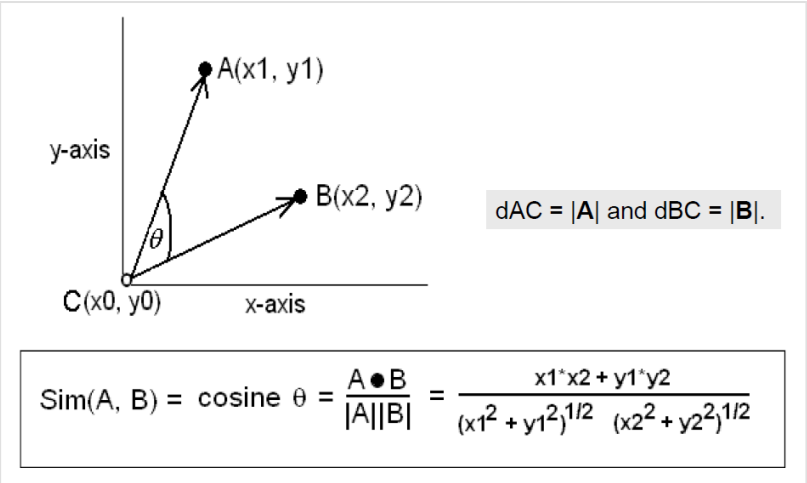
Le moyen le plus simple pour mesurer la distance angulaire est d'utiliser comme mesure le cosinus de l'angle.



Cette représentation des documents sous formes de vecteurs dont les coordonnées sont les poids des différents termes constitue ce que l'on appelle le « modèle vectoriel de Salton ».

De deux dimensions à N dimensions !

Dans la pratique, le modèle vectoriel de Salton ne s'arrête pas à une représentation sur deux axes (donc deux dimensions). On a en fait autant d'axes que de termes indexés ! Mais on sait calculer une distance angulaire, et le cosinus de l'angle, entre des vecteurs définis par des coordonnées dans N dimensions. Il s'agit juste d'un problème de calcul matriciel, le genre de calcul pour lesquels les ordinateurs modernes se révèlent très performants, car ces calculs sont « parallélisables » et « distribuables ». Dans la pratique, le calcul est simplifié car les requêtes contiennent un nombre limité de termes, très inférieur à l'ensemble des dimensions : le calcul peut donc même se faire « à la volée ». Le cosinus de l'angle entre deux vecteurs est donné par la fonction ci-dessous. A.B représente le produit scalaire entre deux vecteurs, facile à calculer quel que soit le nombre de dimensions, et ||A|| représente la norme du vecteur (sa « longueur »).



Calcul du Cosinus de l'angle entre deux vecteurs

Peut-on construire un moteur de recherche en se basant sur ce critère ?

Si l'on utilise seulement ce critère, le moteur ne sera pas efficace, mais combiné à d'autres critères pour créer un algorithme de classement complet, le Cosinus est un critère tout à fait opérationnel. Il était utilisé dans Altavista, et le modèle vectoriel de Salton et son Cosinus ont été utilisés dans de nombreux autres moteurs. Encore aujourd'hui, on retrouve TF*IDF et le Cosinus dans des implémentations du moteur Open Source Lucène (http://fr.wikipedia.org/wiki/Lucene).

Le Cosinus de Salton présente néanmoins quelques inconvénients majeurs, qui conduisent à s'en détourner au profit d'autres méthodes plus efficaces. On peut citer notamment les limites suivantes de cet outil :
- Le modèle vectoriel est un modèle dit « en sac de mots » : il se base sur des statistiques effectuées sans tenir compte de l'ordre des termes, ni des relations sémantiques entre les termes.
- Les pages webs sont souvent composites, et le poids tf*idf n'est pas toujours pertinent sur ce type de page.
- Analyser la similarité cosinus entre des documents très courts (les requêtes) avec des documents très longs donne des résultats souvent mauvais.

Bref, si on peut donc imaginer que des moteurs comme Google ou Bing utilisent des scores calculés selon des méthodes différentes pour contourner les limitations du modèle vectoriel de Salton et de son Cosinus, le modèle n'en reste pas moins intéressant.

Peut-on utiliser le Cosinus de Salton pour améliorer son SEO ?

Dans la pratique, la réponse sera : directement, difficilement, mais indirectement oui.

Il est assez vain de vouloir « optimiser » un contenu en se basant sur les scores obtenus via le cosinus de Salton : l'idée pourrait être de réduire la distance angulaire entre une requête et la page web à optimiser, en ajoutant des termes ou des occurrences de termes. C'est la méthode que nous avions appelé en 2006 « l'alignement sémantique ». Outre le fait que la qualité du texte ainsi « dopé » peut finir par laisser à désirer, l'algorithme de Google est devenu beaucoup plus complexe que cela.

On peut s'en convaincre en testant les résultats remontés par Google sur des requêtes « longue traine » (peu susceptibles d'être influencés par des signaux indépendants de la requête), et en les comparant avec le classement obtenu avec le Cosinus de Salton : les résultats obtenus sont aujourd'hui très différents (c'était moins vrai il y a quelques années).

L'approche peut avoir du sens dans certains cas particuliers, mais souvent le ratio « coût de l'optimisation / bénéfices » est tellement défavorable que la méthode s'avère sans intérêt dans la pratique.

Par contre, bien connaître cet outil peut permettre d'améliorer les performances d'un moteur de recherche interne, ou permettre d'analyser son contenu (en identifiant les documents qui sont «proches » lexicalement). On peut s'en servir dans des algorithmes de calcul de maillage interne par exemple, pour identifier les pages intéressantes à relier entre elles. Même s'il existe des méthodes plus modernes, et plus efficaces, le modèle vectoriel de Salton et la similarité Cosinus sont connus depuis 40 ans maintenant : il existe donc de nombreux outils et de nombreuses librairies qui « embarquent » les scripts qui permettent de l'implémenter sur de nombreuses plateformes et dans de nombreux langages informatiques.

Indexation des syntagmes, indexation des concepts, indexation des objets et des entités nommées : le modèle vectoriel de Salton est-il déjà enterré ?

Depuis les années 70, les outils utilisés en « recherche d'information » ont considérablement évolué. Dans Google, on a vu apparaître des comportements qui démontrent que le moteur indexe non seulement des termes isolés, mais aussi des syntagmes (expressions, groupes nominaux), des entités ou des objets (noms de personnes, de sociétés, dates…), des relations entre entités (cf. le Knowledge Graph) et plus récemment des « concepts » avec Hummingbird.

Cela signifie-t-il que le modèle vectoriel de Salton est mort et enterré ? Pas tout à fait. Ces nouvelles approches sont des « sophistications » successives du modèle initial de Salton. En réalité il est rare que dans un moteur de recherche, une approche nouvelle remplace totalement une approche plus ancienne jugée efficace. Un algorithme de recherche est un empilement de scores, se basant sur de multiples signaux, et dont l'objectif est de fournir un score final capable de déterminer le classement des URL dans les pages de résultat.

Bref, même en 2014, on peut toujours voir le Cosinus de Salton comme un composant historique des algorithmes des moteurs de recherche, tellement incontournable qu'il est encore présent dans des moteurs modernes. Tant pis si on ne sait pas vraiment s'il est utilisé par Google : comprendre comment il a été implémenté dans des moteurs de recherche opérationnels (en combinaison avec d'autres critères) fournit beaucoup d'enseignements sur la nature réelle des algorithmes des moteurs de recherche.

Bibliographie

L'article fondateur de Gerard Salton
G. Salton, A. Wong, and C. S. Yang (1975), "*A Vector Space Model for Automatic Indexing*," Communications of the ACM, vol. 18, nr. 11, pages 613–620.
http://www.cs.uiuc.edu/class/fa05/cs511/Spring05/other_papers/p613-salton.pdf

Tutorial d' E. Garcia sur le modèle vectoriel de Salton et le cosinus
http://hi.baidu.com/dfeexiao/item/de6598d9d4e6e54cddf9be69

Tutoriel complet avec des exemples de code en Python sur le modèle vectoriel de Salton
http://pyevolve.sourceforge.net/wordpress/?p=1589

Modèle vectoriel et mythes SEO (par E. Garcia)
http://irthoughts.wordpress.com/2008/04/21/vector-space-models-and-search-engines/

Un outil en ligne pour tester le calcul d'une similarité cosinus
http://www.appliedsoftwaredesign.com/archives/cosine-similarity-calculator/

Philippe YONNET , ***Directeur de l'agence Search-Foresight / Groupe MyMedia. Président de l'association SEO Camp (http://www.seo-camp.org/)***