République Algérienne Démocratique Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université d'Ibn Khaldoun – Tiaret
Faculté des Mathématiques et de l'Informatique
Département d'Informatique

Thème

Développement d'un outil de recherche d'information sur web

Pour l'obtention du diplôme de Master II

Spécialité : Réseaux et télécommunications

Réalisé par : FACTOUR Abdelhak

LEBOUAHLA Mounir

Dirigé par : LAKHDARI Aicha

Année universitaire 2018-2019

Remerciements

Je remercie, tout d'abord, Dieu tout puissant de m'avoir donné le courage, et la patience d'achever ce modeste travail.

Je tiens tout d'abord à exprimer ma profonde gratitude à Mme Lakhdari aicha mon encadreur.

Au département d'Informatique université de Tiaret, pour sa présence scientifique et humaine, et l'honneur qu'il m'on fait en acceptant de nos encadrer.

Avec un intérêt constant et une grande compétence ainsi pour l'intérêt qu'ils ont bien voulu porter à nos travail.

Mes respects et ma gratitude vont également aux membres du jury qui m'ont fait l'honneur de juger ce travail et qui par leur disponibilité, leurs observations et leurs rapports m'ont permis d'enrichir notre travail.

Je remercie toutes les personnes qui ont participé de manière directe ou indirecte à la concrétisation de ce travail.

Table des matières

INTRODUCTION GÉNÉRALE

Introduction générale :

La RI est une tâche très populaire, à laquelle tous les internautes font appel quotidiennement dès qu'ils utilisent un moteur de recherche. Ceux-ci appartiennent à plusieurs familles : il y a, bien sûr, les moteurs généralistes (Google, Bing, Yahoo!, Baidu en Chine...) qui servent à s'orienter sur l'ensemble du Web, mais la plupart des sites importants (notamment tous les sites marchands ou institutionnels) disposent aussi d'un moteur interne permettant de naviguer à l'intérieur de leurs pages. Tout internaute sollicite donc quotidiennement, parfois sans le savoir, plusieurs moteurs de recherche. Des systèmes de recherches ont aussi intégrés au cœur même de chaque ordinateur, pour aider l'utilisateur à fouiller dans son disque dur à la recherche d'un fichier ou d'un mail mal rangé. Enfin, la RI existait déjà avant même l'invention du Web, dans le domaine des "sciences de la documentation". Elle était dans ce cadre cantonnée aux archives et aux bibliothèques, pionnières en matière d'indexation et de requêtage de corpus de textes numérisés. Plutôt que de moteurs de recherche, on parlait alors de "logiciels documentaires". Les techniques utilisées pour construire un programme de RI dans ces différents contextes peuvent varier, mais restent assez homogènes.

Tout système de recherche se compose d'un modèle de représentation des documents, d'un modèle de représentation des requêtes et d'un modèle de correspondance (pertinence) entre les représentations des requêtes et des documents. Le premier modèle (appelé aussi langage d'indexation) exprime le contenu informationnel des documents dans un formalisme de représentation des connaissances. Le modèle de représentation des requêtes exprime le contenu informationnel du besoin en information de l'utilisateur. Dans la plupart des approches existantes, pour des raisons de simplicité et de cohérence, ce modèle utilise le même formalisme de représentation des connaissances que le modèle de représentation des documents. Le modèle de correspondance quantifié le degré de similarité entre une représentation d'une requête et les représentations des documents.

Pour une requête soumise à notre système de recherche d'informations (SRI), chaque document est soit pertinent ou soit non pertinent.

En réalité, la notion de pertinence est graduelle et nous attendons plus de notre SRI qu'il ordonne du plus au moins pertinent les documents plutôt qu'il les classe suivant un simple critère. C'est particulièrement sensible pour les corpus volumineux, ou seuls les premiers résultats proposés sont réellement consultés par les utilisateurs. Notre travail s'articule autour du modèle d'appariement ainsi il consiste à réaliser un SRI et plus précisément un modèle de recherche d'information vectoriel (modèle d'espace vectoriel). Le modèle vectoriel est le premier à intégrer

un élément fondamental : la capacité d'ordonner les documents restitués selon un critère de pertinence. Nos expérimentations sont menées sur une collection de documents locale (localhost) et à distance sur le web.

Ce présent mémoire est composé de trois chapitres :

Dans le premier chapitre décrit le processus de RI, les objets manipulés dans ce processus, ainsi que les différents traitements pouvant être effectués sur les données textuelles.

Dans le second chapitre, nous avons présenté le modèle vectoriel plus précisément sa définition, son principe général et sa façon de faire correspondre les documents à une requête donnée par un utilisateur.

Le troisième chapitre présente les outils de développement et les expérimentations réalisés pour récupérer un résultat pertinent de documents.

CHAPITRE I

Recherche d'information (RI)

I.1. Introduction

La Recherche d'Information (RI) peut être définie comme une activité dont la finalité est de localiser et de délivrer un ensemble de documents à un utilisateur en fonction de son besoin en informations. Le défi est de pouvoir, parmi le volume important de documents disponibles, trouver ceux qui correspondent au mieux à l'attente de l'utilisateur.

L'opération de la RI est réalisée par des outils informatiques appelés Systèmes de Recherche d'Information (SRI), ces systèmes ont pour but de mettre en correspondance une représentation du besoin de l'utilisateur (requête) avec une représentation du contenu des documents au moyen d'une fonction de comparaison (ou de correspondance). L'essor du web a remis la RI face à de nouveaux défis d'accès à l'information, il s'agit cette fois de retrouver une information pertinente dans un espace diversifié et de taille considérable. Ces difficultés ont donné naissance à une nouvelle discipline appelée Recherche d'Information sur le Web.

Dans ce chapitre, nous nous intéressons au processus de recherche d'informations. Après avoir donné les définitions, nous étudions l'architecture générale des systèmes de recherche d'informations ainsi que les trois principaux modèles : booléen, vectoriel et probabiliste. Les capacités d'un tel système sont étroitement liées à la représentation des documents qui est utilisée, soulignant l'importance des éléments pris en compte lors de l'indexation. Nous prêtons une attention particulière à la notion de pertinence, qui se trouve au centre du processus de recherche d'informations.

I.2. Définitions

Plusieurs définitions de la recherche d'information ont vu le jour dans ces dernières années, nous citons :

 la Recherche d'Information peut se définir comme : « Action, méthodes et procédures ayant pour objet d'extraire d'un ensemble de documents les informations voulues. Dans un sens plus large, toute opération (ou ensemble d'opérations) ayant pour objet la recherche, la collecte et l'exploitation d'informations en réponse à une question sur un sujet précis » d'après l'AFNOR[1].

 La recherche d'information est une activité dont la finalité est de localiser et de délivrer des granules documentaires à un utilisateur en fonction de son besoin en informations (1).

[1] www.afnor.org

 La recherche d'information est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche et la sélection d'information (2).

 La recherche d'information est une discipline de recherche qui intègre des modèles et des techniques dont le but est de faciliter l'accès à l'information pertinente pour un utilisateur ayant un besoin en information (3).

Ces notions se confondent dans les systèmes actuels du fait notamment de la dématérialisation massive des documents et de l'intérêt toujours grandissant du Web. L'utilisateur peut donc indifféremment réaliser ces types de recherche et ce de façon transparente. Il y a donc un accès direct au contenant et au contenu.

La plupart des activités quotidiennes font appel à la RI et ce quel que soit le contexte (professionnel, loisirs…) pour combler un manque en information, vérifier voire valider une information…

I.3. Concepts de base de la RI

Plusieurs concepts clés s'articulent autour de la dentition d'un système de RI (4) :

I.3.1 Collection de documents

La collection de documents (ou corpus) constitue l'ensemble des informations (des documents) exploitables et accessibles.

I.3.2 Document

Le document constitue l'information élémentaire d'une collection de documents. L'information élémentaire, appelée aussi granule de document, peut représenter tout ou une partie d'un document.

I.3.3 Besoin en information

Cette notion est souvent assimilée au besoin de l'utilisateur. Ingwersen (5) a défini trois types de besoins utilisateur :

– *Besoin vérificatif* : l'utilisateur cherche à vérifier le texte avec les données connues qu'il possède déjà. Il recherche donc une donnée particulière, et sait même souvent comment y accéder. La recherche d'un article sur Internet à partir d'une adresse connue serait un exemple d'un tel besoin. Un besoin de type vérificatif est dit stable, c'est-à-dire qu'il ne change pas au cours de la recherche.

– *Besoin thématique connu* : l'utilisateur cherche à clarifier, à revoir ou à trouver de nouvelles informations dans un sujet et domaine connus. Un besoin de ce type peut être stable ou variable ; il est très possible en effet que le besoin de l'utilisateur s'affine au cours de la recherche.

– *Besoin thématique inconnu* : pour ce type de besoins, l'utilisateur cherche de nouveaux concepts ou de nouvelles relations hors des sujets ou domaines qui lui sont familiers. Le besoin est intrinsèquement variable et est toujours exprimé de façon incomplète.

I.3.4 Requête

La requête est l'expression du besoin en information de l'utilisateur. Elle représente l'interface entre le système de RI et l'utilisateur. Divers types de langages d'interrogation sont proposés dans la littérature. Une requête est un ensemble de mots clés, mais elle peut être exprimée en langage naturel, booléen ou graphique.

I.3.5 Pertinence

La pertinence est une notion fondamentale dans le domaine de la RI. La pertinence peut être définie comme la correspondance entre un document et une requête, ou encore une mesure d'informativité du document à la requête (6). Ces différents critères ont amené à la catégorisation de la pertinence utilisateur principalement en 5 classes de pertinence (7) :

– la *pertinence algorithmique* (ou système) : souvent présentée par un score de l'adéquation du contenu des documents vis-à-vis de celui de la requête. Pour mesurer cette adéquation, le système de RI procède au calcul du degré de similitude du document et de la requête en se basant sur les représentations internes de chacun de ceux-ci. Le but de tout système de RI est de rapprocher la pertinence algorithmique calculée par le système aux jugements de pertinence donnés par des vrais utilisateurs.

– la *pertinence thématique* : traduit le degré d'adéquation de l'information retrouvée au thème évoqué par le sujet de la requête. C'est la mesure la plus utilisée dans les moteurs de recherche classiques.

– la *pertinence cognitive* : représente la relation entre l'état de la connaissance intrinsèque de l'utilisateur et l'information portée par les documents telle qu'interprétée par l'utilisateur, cette pertinence se caractérise par une dynamique qui permet d'améliorer la connaissance de l'utilisateur via l'information renvoyée le long de sa recherche.

– la *pertinence situationnelle* : est vue comme l'utilité de l'information retrouvée par rapport à la tâche ou le problème posé par l'utilisateur.

– la *pertinence motivationnelle* (ou affective) : décrit la relation entre les intentions, les buts et les motivations de la recherche tels que fixés par l'utilisateur d'une part et les informations retrouvées d'autre part.

I.4. Processus générale de la RI:

Le processus de RI qui permet, à partir d'une requête, d'ordonnancer les documents est appelé "processus en U". Il est décomposé en trois principales étapes, illustrées dans la Figure I-1 et détaillées ci-dessous.

• L'*indexation* des documents et des requêtes de l'utilisateur.

• L'*appariement* document-requête.

• La *reformulation* de la requête.

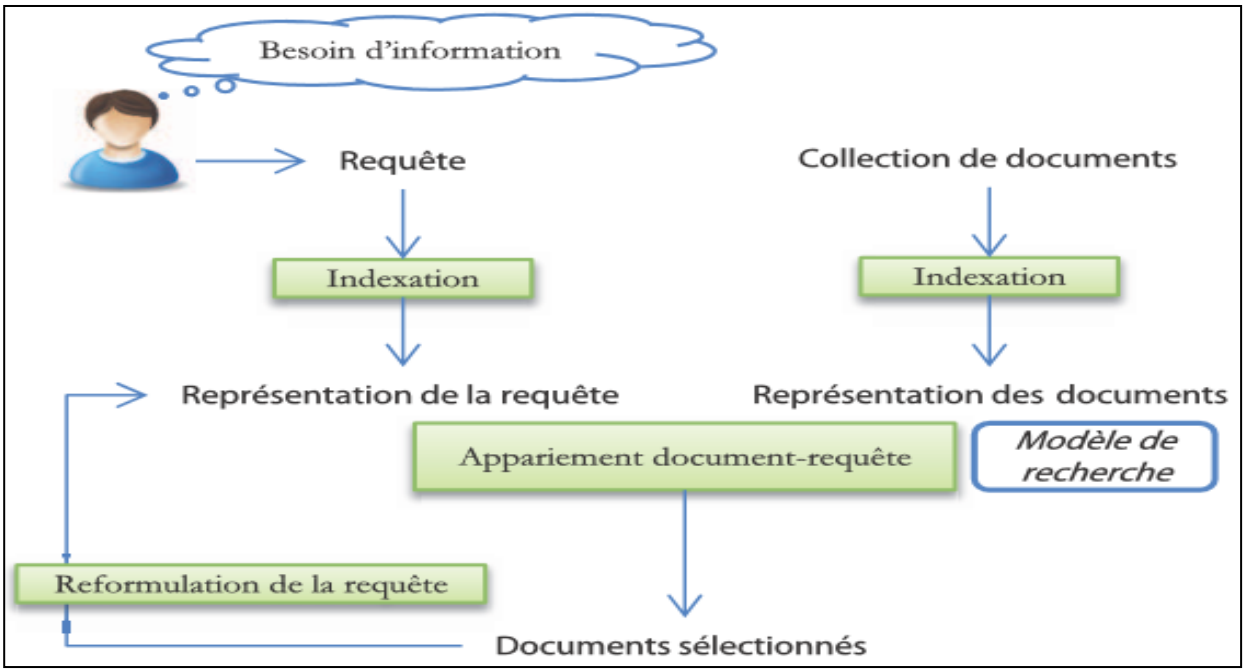

Figure I-1: Processus en U de la recherche d'information

I.4.1 Indexation

Les documents à leur état brut sont difficiles à exploiter tels quels lors de la phase de recherche. Ainsi, l'objectif principal de cette étape est de fournir des représentations des documents et des requêtes facilement exploitables par la machine dans la phase de recherche. Cette représentation est souvent une liste pondérée de mots-clés significatifs que l'on nomme descripteurs du document (ou de la requête) (8). L'indexation peut être :

• *Manuelle :* la représentation du document est réalisée par un expert qui identifie les termes les plus représentatifs du document.

•*Automatique :* le processus d'indexation est entièrement informatisé. Il repose sur une démarche algorithmique qui traite chaque terme selon un processus défini : extraction, suppression des mots vides, normalisation et pondération...

• *Semi-automatique :* est une combinaison des deux précédentes approches où le choix final des termes à indexer revient à l'expert.

A la fin de cette étape, les documents sont représentés dans des fichiers index qui stockent la cartographie des couples terme-document en y associant un poids. La formule de pondération la plus utilisée est celle basée sur la fréquence des termes dans les documents, appelée TF-IDF (9). L'intuition de cette pondération est de favoriser les termes qui sont à la fois fréquents dans le document et peu fréquents dans la collection. Cette dernière condition est basée sur les propriétés de la loi de Zipf (10) qui étudie la distribution des termes dans une collection de documents.

L'indexation automatique (11) regroupe un ensemble de traitements automatisés sur un document comme :

• *l'extraction des mots :* Ce processus consiste à analyser le texte d'un document afin d'extraire ses mots en reconnaissant les espaces de séparation des mots, les ponctuations, etc.

• *l'élimination des mots vides :* Un document contient souvent des mots non significatifs appelés *mots vides* (pronoms personnels, prépositions).

L'élimination de ces mots se fait à l'aide d'une liste prédéfinie de mots vides ou en supprimant les mots ayant une fréquence dépassant un certain seuil. Éliminer les mots vides permet de réduire la taille de l'index, gagner en espace mémoire et optimiser le temps d'exécution.

• *la lemmatisation :* Ce traitement consiste à radicaliser les mots restants, c'est à dire réduire les mots à leur forme canonique par exemple, toutes les formes d'un verbe sont regroupées à l'infinitif, tous les mots au pluriel sont ramenés au singulier, etc. Grâce à la lemmatisation, les documents contenant différentes formes d'un même terme auront les mêmes chances d'être restitués ce qui améliore la capacité d'un SRI à retrouver les documents pertinents. Parmi les méthodes utilisées pour la lemmatisation on peut citer l'algorithme de Porter (12) pour les textes en anglais et la troncature (13) pour les autres langues (Français, Italien, Allemand).

• *la pondération :* Les termes d'un document n'ont pas souvent la même importance. Un terme qui apparaît dans la majorité des documents de la collection aura moins d'importance qu'un terme qui existe dans quelques documents seulement. Plusieurs fonctions de pondération de termes ont été proposées dans la littérature. La plupart de ces fonctions combinent des variantes des facteurs *TF (Term Frequency)* et *IDF (Inverse Document Frequency)* qui mesurent un poids local (dans le document) et global (dans la collection) d'un terme (14) .

La mesure *TF-IDF*, font intervenir deux facteurs : fréquence du terme *t* dans le document *d*, notée *TF* (*Term Frequency)*, et la fréquence inverse du document, notée *IDF* (*Inverse Document Frequency*)

La formule du *TF-IDF* est donnée par le produit des deux fonctions *TF* et *IDF* comme suit: $TF - IDF_{t,d} = TF_{t,d} \times IDF_t$

1. TF **(Term Frequency)** : ce facteur prend en compte le nombre d'occurrence d'un terme dans un document. L'idée derrière cette mesure est que plus un terme est fréquent dans un document plus il est important. Elle représente une pondération locale d'un terme dans un document.

2. *IDF* **(Inverse Document Frequency)** : ce facteur mesure la fréquence d'un terme dans toute la collection, c'est la pondération globale.

I.4.2 Appariement document-requête

Le processus d'appariement met en relation la collection de documents, indexée au préalable, avec la requête, également prétraitée, afin d'identifier les documents pertinents. Cette étape permet au SRI de retourner une liste de documents à l'utilisateur.

Dans le processus d'appariement, le système calcule un score de correspondance entre la représentation de chaque document et celle de la requête. Ce score peut être binaire (pertinent ou non pertinent) ou multivalué pour exprimer un degré de pertinence système. La pertinence système est calculée à partir d'une fonction de similarité appelée RSV (Q, *D*) (Retrieval Status Value) où Q est une requête et D un document. Pour une requête donnée, le système retourne des documents en ordre décroissant du score de pertinence. L'appariement *document-requête* repose sur un cadre théorique défini par un modèle de recherche d'information. Une taxonomie des modèles (Figure I-2) a été présentée par Baeza-Yates (15) et présente quatre familles principales. Les modèles reposant sur le texte des documents

(modèles de RI classiques et modèles basés sur le texte semi-structuré), les liens entre documents (modèles orientés web) et les documents multimédia (recherche d'images, de musiques, d'audio ou de vidéos).

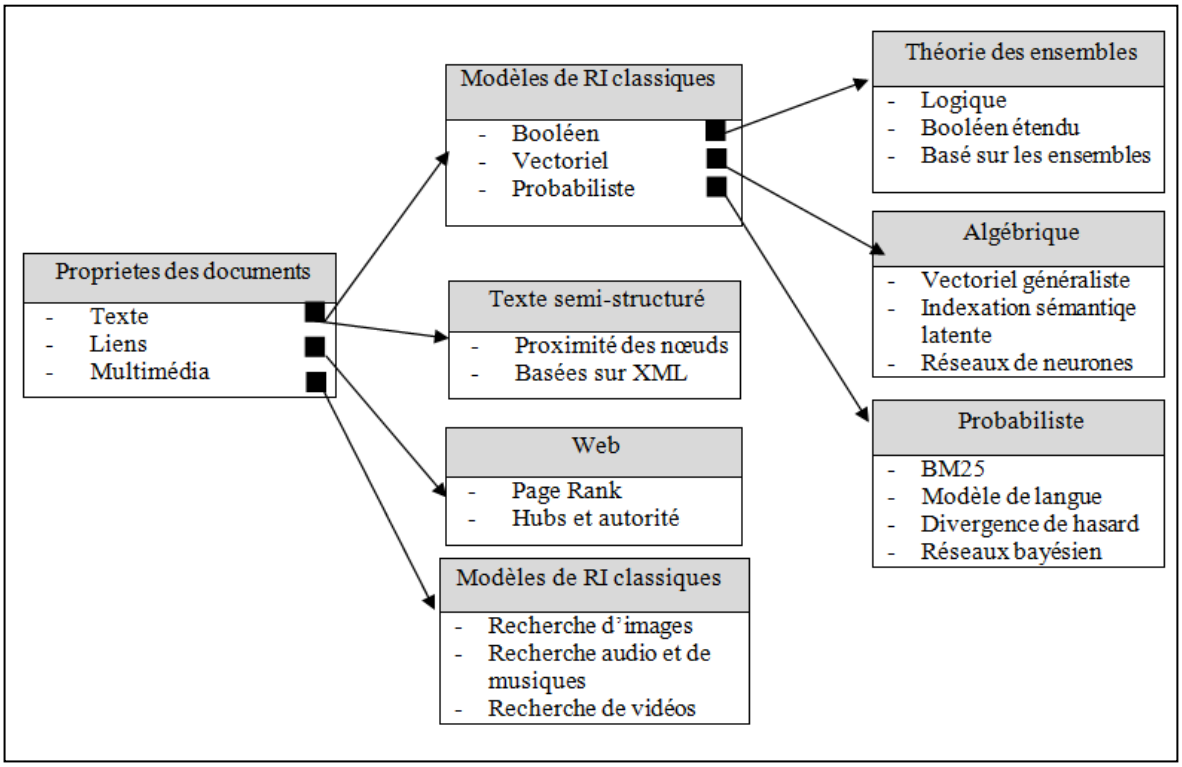

Figure I-2: Taxonomie des modèles de RI.

Un modèle de recherche d'informations est un cadre de calcul qui, à partir d'une représentation des documents et une représentation de la requête, détermine la relation ou le degré de similitude entre le document et la requête.

Ici, nous présentons les trois grands modèles utilisés en recherche d'informations : booléen, vectoriel et probabiliste :

I.4.2.1 Modèle booléen

Le modèle booléen (16) est le modèle le plus ancien dans la recherche d'information. Il est basé sur la théorie des ensembles et l'algèbre de Boole. Le document est représenté par un ensemble de termes. La requête est représentée sous forme d'une expression logique composée de termes reliés par des opérateurs logiques ***ET***, ***OU***, ***NON***. L'appariement (RSV) entre une requête et un document est un appariement exact, autrement dit si un document implique au sens logique de la requête alors le document est pertinent. Sinon, il est considéré non pertinent.

Par conséquent, le score de similarité entre un document *d* et une requête *q* est inclus dans l'ensemble {0, 1}:

$$\textbf{RSV}\ (\boldsymbol{d}, \boldsymbol{q}) = \begin{cases} 1 & \text{si d appartient à l'ensemble décrit par q} \\ 0 & \text{Sinon} \end{cases}$$

Le modèle booléen a des avantages qui sont présentés par :

- Le modèle de recherche booléen est reconnu pour sa force pour faire une recherche très restrictive et obtenir, pour un utilisateur expérimenté, une information exacte et spécifique.

- La simplicité du modèle le rend aisément compréhensible pour un utilisateur.

- L'efficacité du modèle est due aux spécialistes qui ont explorés le corpus avec une bonne connaissance du vocabulaire.

- La formulation des requêtes devient vite laborieuse quand la requête se fait précise.

Ce modèle n'a pas seulement d'avantages, il a aussi des inconvénients qui sont les suivants :

- La représentation binaire d'un terme dans un document est peu informative, car elle ne renseigne ni sur la fréquence du terme dans le document ni sur la longueur de document, qui peuvent constituer des informations importantes pour la RI.
- Les documents retournés à l'utilisateur ne sont pas ordonnés selon leur pertinence.

- L'impossibilité de rendre compte d'une correspondance partielle d'un document à une requête.

- Il est difficile pour les utilisateurs de formuler de bonnes requêtes. Par conséquent, l'ensemble des documents trouvés est souvent trop grand, pour les requêtes courtes, ou complètement vide dans le cas de requêtes longues.
- Les tests effectués sur des collections d'évaluation standards de RI ont montré que les systèmes booléens sont d'une efficacité de recherche inférieure.

I.4.2.2 Modèle vectoriel

Initialement proposé par Salton et implémenté dans le système SMART (17). La pertinence d'un document vis-à-vis d'une requête est définie par des mesures de distance dans un espace vectoriel. Le modèle vectoriel représente les documents et les requêtes par des vecteurs d'un espace à n dimensions, les dimensions étant constituées par les termes du vocabulaire d'indexation (Figure I-3).

Les coordonnées d'un vecteur document sont les poids des termes d'index dans ce document.

$$D_i = w_{i1}, w_{i2}, w_{i3} \ldots w_{in} \qquad \text{pour } i = 1, 2 \ldots m$$

Où w_{ij} est le poids du terme t_j dans le document D_i,

m est le nombre de documents dans la collection,

n est le nombre de termes d'indexation.

On représente aussi la requête par un vecteur de mots-clés défini dans le même espace vectoriel que le document. $Q = w_{Q1}, w_{Q2}, \ldots w_{Qn}$,

Où w_{Qj} est le poids de terme t_j dans la requête Q.

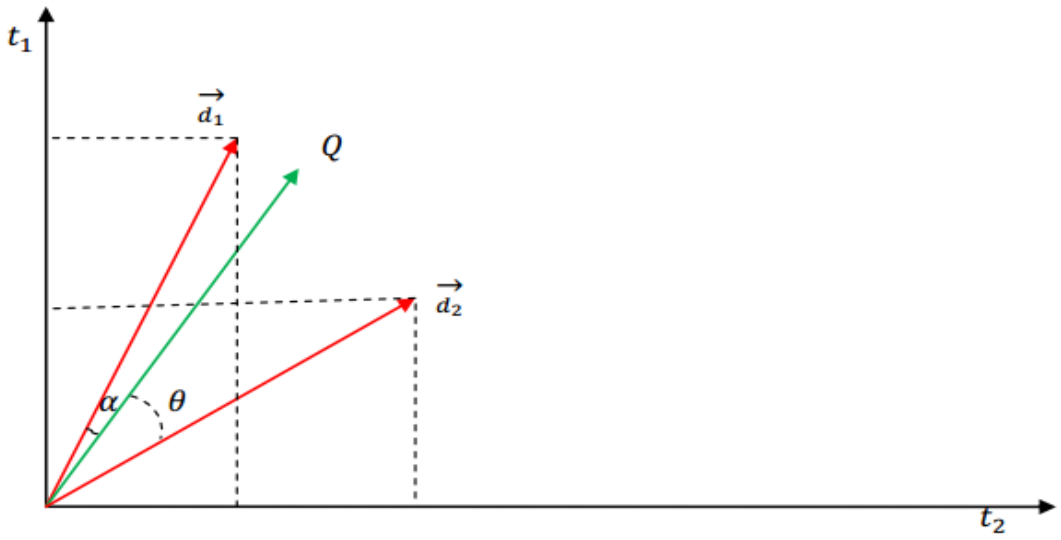

Figure I-3: Représentation algébrique des documents et des requetes dans l'espace des termes à deux dimensions

L'appariement document-requête dans le modèle vectoriel, consiste à trouver les vecteurs documents qui s'approchent le plus de vecteur de la requête. Cet appariement est obtenu par l'évaluation de la distance entre les deux vecteurs. Plusieurs mesures de similarité ont été définies (18) **,** dont les plus courantes sont décrites ci-dessous.

- **Le produit scalaire :**

$$RSV(q_i, d_j) = \sum_{k=1}^{M} w_{ki} \,.\, w_{kj}$$

- **La mesure de Jaccard :**

$$RSV(q_i, d_j) = \frac{\sum_{k=1}^{M} w_{ki} \cdot w_{kj}}{\sum_{k=1}^{M} w_{ki}^2 + \sum_{k=1}^{M} w_{kj}^2 - \sum_{k=1}^{M} w_{ki}^2 \cdot w_{kj}}$$

La mesure cosinus :

$$RSV(q_i, d_j) = \frac{\sum_{k=1}^{M} w_{ki} \cdot w_{kj}}{\sqrt{\sum_{k=1}^{M} w_{ki}^2} \cdot \sqrt{\sum_{k=1}^{M} w_{kj}^2}}$$

Plus les vecteurs sont similaires, plus l'angle formé est petit, et plus le cosinus de cet angle est grand. A l'inverse du modèle booléen, la fonction de correspondance évalue une correspondance partielle entre un document et une requête, ce qui permet de retrouver des documents qui ne reflètent pas la requête qu'approximativement. Les résultats peuvent donc être ordonnés par ordre de pertinence décroissante (19).

Le modèle vectoriel a des avantages (8) qui sont présentés par :

- La pondération améliore les résultats de recherche.
- Le modèle permet une correspondance partielle ou approximative entre les documents et les requêtes (*best match*).
- La mesure de similarité permet d'ordonner les documents selon leur pertinence vis à vis de la requête.

Ce modèle n'a pas seulement d'avantages, il a aussi des inconvénients qui sont les suivants :

- L'inconvénient majeur de modèle vectoriel est qu'il repose sur l'hypothèse de l'indépendance des termes d'indexation, or ces termes dans les documents sont souvent sémantiquement liés (16).
- Il comporte également plusieurs limitations qui furent, pour certaines, corrigées par des affinements du modèle (20).
- Dans un texte l'ordre des mots, les synonymes, la morphologie des contenus ne sont pas pris en compte (20).

I.4.2.3 Modèle probabiliste

Ce modèle est fondé sur le calcul de la probabilité de pertinence d'un document pour une requête (21) (22) (11)…Le principe de base consiste à retrouver des documents qui ont en même temps une forte probabilité d'être pertinents, et une faible probabilité d'être non pertinents.

Etant donné une requête utilisateur Q et un document D, il s'agit de calculer la probabilité de pertinence du document pour cette requête.

Le score de pertinence d'un document d par rapport à la requête q est estimé comme suit :

$$RSV(q, d_j) = \frac{P(P|d_i)}{P(\bar{P}|d_i)}$$

Où $P(P|d_i)$ et $P(\bar{P}|d_i)$: La probabilité q'un document d_i soit pertinent (P) vis-à-vis de la requête q (respectivement non pertinent $P(\bar{P}|d_i)$).

Ces probabilités sont estimées par de probabilités conditionnelles selon qu'un terme de la requête est présent, dans un document pertinent ou dans un document non pertinent. Cette mesure de similarité entre la requête et les documents peut se calculer par différentes formules. Ce modèle a donné lieu à de nombreuses extensions. Il est à l'origine du système OKAPI qui est l'un des systèmes les plus performants selon les compagnes d'évaluation TREC[2]. L'inconvénient majeur de ce modèle est que les calculs des probabilités sont complexes et que l'indépendance des variables n'est pas toujours vérifiée voir pas prise en compte (23).

Ces modèles ont une base théorique saine (24) et se sont montrés particulièrement performants dans TREC.

I.4.3 Reformulation de la requête

La reformulation du besoin en information est l'étape qui permet de redéfinir le besoin de l'utilisateur au fur et à mesure de la session de recherche.

Cette étape peut être effectuée :

- Manuellement, dans le cas où l'utilisateur soumet lui-même une nouvelle requête.
- De façon automatique, lorsque le système de RI s'appuie sur les termes importants dans les documents les plus pertinents ou visités par l'utilisateur qui sont réutilisés.

L'une des stratégies de reformulation de requêtes est celle qui est dirigée par l'utilisateur. Le principe de cette stratégie est de construire une nouvelle requête à partir de la structure des documents jugés par l'utilisateur : c'est ce que l'on appelle la réinjection de pertinence « relevance feedback » (25) (26) (27).

[2] WWW.TREC.com

La reformulation est un processus évolutif et interactif. Son principe fondamental est d'utiliser la requête initiale pour amorcer la recherche, puis modifier celle-ci à partir des jugements de pertinence et/ou de non-pertinence de l'utilisateur dans le but de repondérer les termes de la requête initiale, ou y ajouter (respectivement supprimer) d'autres termes contenus dans les documents pertinents (respectivement non pertinents). La nouvelle requête obtenue à chaque itération de feedback, permet de corriger la direction de la recherche dans le sens des documents pertinents. En effet, la simple comparaison du contenu de la requête et des documents de la base ne permet pas d'avoir tous les documents correspondant à une requête donnée. Il reste toujours des documents pertinents non restitués, car ne contenant pas les termes de la requête (23).

I.5. Evaluation de SRI

La démarche de validation en RI (23) se base sur l'évaluation expérimentale des performances du modèle ou du système proposé. L'évaluation des performances d'un modèle de RI, permet de paramétrer le modèle, d'estimer l'impact de chacune de ses caractéristiques et de fournir des éléments de comparaison entre modèles. Cette évaluation peut porter sur plusieurs critères : le temps de réponse, la pertinence, la qualité et la présentation des résultats, etc.

Le critère le plus important est celui qui mesure la capacité du système à satisfaire le besoin en information de l'utilisateur, c'est à dire la pertinence.

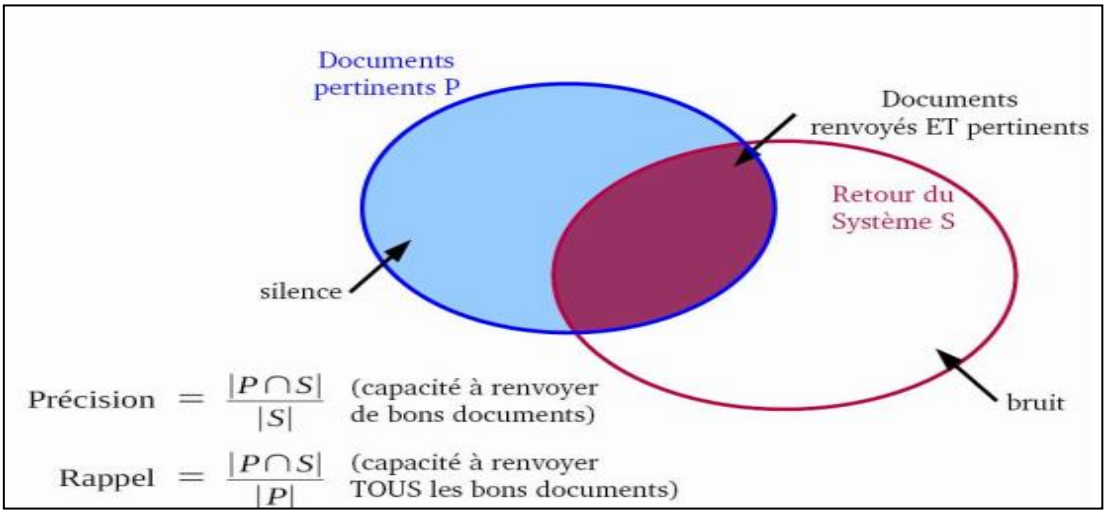

Figure I-4: Précision et rappel

Deux facteurs permettent d'évaluer ce critère. Le premier est le taux de rappel, il mesure la capacité du système à sélectionner tous les documents pertinents. Le second est le taux de

précision, il mesure la capacité du système à rejeter tous les documents non pertinents. On calcule :

$$\textbf{Rappel} = \frac{\text{Nombre de documents pertinents restitués}}{\text{Nombre de documents pertinents}}$$

$$\textbf{Précision} = \frac{\text{Nombre de documents pertinents restitués}}{\text{Nombre de documents restitués}}$$

Ce type de mesure est effectué sur des collections de tests. Une collection de tests est composée d'un ensemble de documents, d'un ensemble de requêtes et d'un ensemble de jugements de pertinence. L'initiative la plus importante actuellement pour la construction de collections de tests est sans conteste TREC (26). TREC est plus qu'une collection de tests, c'est un programme d'évaluation des SRI, initié par le NIST aux USA. TREC fournit une plate-forme comportant des collections de tests, des tâches spécifiques et des protocoles d'évaluation pour chaque tâche, pour l'évaluation et la comparaison d'expérimentations sur des collections volumineuses de textes.

I.6. Conclusion :

Nous avons présenté dans ce chapitre les principales notions et concepts de la recherche d'information. Nous y avons développé les principales étapes d'un processus de recherche d'information, à savoir, la représentation ou indexation de l'information, la comparaison de l'information et du besoin en information. Les principaux modèles existants, ainsi que les différentes méthodes d'évaluation des performances des systèmes de recherche d'information. Le chapitre suivant est consacré à la présentation du modèle algébrique qui est le modèle vectoriel.

CHAPITRE 2

Le modèle vectoriel

1. Introduction

L'indexation choisit les termes pour représenter le contenu d'un document ou d'une requête, le modèle permet de donner une interprétation des termes choisis pour représenter le contenu d'un document. Étant donné un ensemble de termes pondérés issus de l'indexation, le modèle remplit deux fonctions :

— La première est de créer une représentation interne pour un document ou pour une requête basée sur ces termes,

— La seconde est de définir une méthode de comparaison entre une représentation de document et une représentation de requête afin de déterminer leur degré de correspondance (ou similarité). Le modèle joue un rôle central dans la RI. C'est lui qui détermine le comportement clé de notre SRI. De nombreux modèles existent dans la litterature. Dans la suite, le modèle, qui sert de base à notre modèle de RI, est un modèle algébrique utilisant la représentation tabulaire du modèle vectoriel.

2. Principe du modèle vectoriel

Le principe fondamental est qu'il doit y avoir correspondance optimale entre les termes de la requête et ceux contenus dans les documents indexés. Le moteur compare l'ensemble des termes présents dans la requête à l'ensemble des radicaux des termes présents dans la base de données, effectue une mesure de leur proximité et délivre une liste de réponses triées suivant ce critère de proximité. Ainsi, les documents correspondant le mieux à la requête sortent au début de la liste. Il s'agit donc d'un calcul de proximité entre la requête et les descriptions, que l'on appelle pertinence (Relevance en anglais). Une des façons les plus utilisées pour calculer la pertinence s'appuie sur le modèle vectoriel. Chaque document et chaque requête sont caractérisés par la liste de ces valeurs numériques qui forment un vecteur.

Ils peuvent donc être traités mathématiquement comme des vecteurs dans l'espace. [22] L'idée au cœur de notre approche est de se servir d'un nombre m de documents dits documents-références. Pour chaque document de la collection, on calcule alors sa similarité, qui est un score, à chacune de ces références selon le modèle de similarité imposé par le SRI utilisé (chaque document de la collection joue le rôle de la requête).

Les m valeurs obtenues sont rassemblés dans un vecteur qui caractérise désormais chaque document. Ce processus, éventuellement coûteux, se fait bien sûr hors-ligne. Les requêtes subissent le même traitement, en ligne, toutefois.

Les requêtes et les documents se trouvent donc représentés par des vecteurs de dimensions m. La distance entre une requête et les documents de la collection se calcule ensuite de manière similaire à ce qui se fait dans les modèles vectoriels, et ce indépendamment du modèle utilisé pour construire ces vecteurs.

La recherche de documents pertinents pour une requête se fait donc de manière indirecte, par le bais des proximités avec les documents-références. La proximité finale peut donc être dite de second ordre. Cela implique aussi qu'un document puisse être jugé pertinent pour une requête même s'il ne contient aucun mot de cette requête. Bien entendu, pour que cette technique soit intéressante d'un point de vue calculatoire, il faut que m < D, où D est le nombre de documents de la collection.

3. Interprétation du Modèle vectoriel

Le modèle vectoriel est une représentation mathématique du contenu d'un document, selon une approche algébrique. Ce modèle se base sur une formalisation géométrique. En effet les documents et les requêtes sont représentés dans le même espace, défini par un ensemble de dimensions, chaque dimension représente un terme d'indexation.

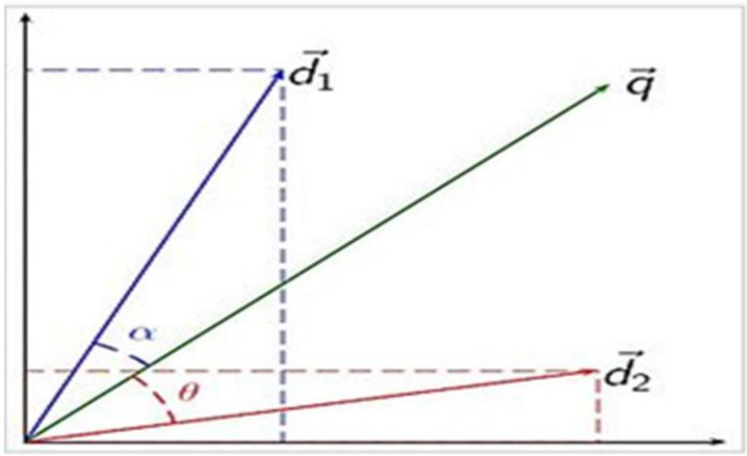

Figure 3.1 : représentation de deux documents (d1et d2) et d'une requête (q).

Dans un espace vectoriel.la proximité de la requête aux documents est représentée par les angles α et entre les vecteurs.

Dans un espace vectoriel, il est possible de calculer la distance entre le vecteur qui correspond à la requête et ceux qui correspondent à des descriptions. Il s'agit d'une mesure de similarités, qui s'appuie souvent sur la valeur du cosinus entre deux vecteurs. Une valeur 1 coïncide avec une correspondance exacte. Ensuite, la valeur diminue à mesure que les vecteurs s'éloignent : 0 correspond à un angle de 90° et -1 à un angle plat (vecteurs diamétralement opposés).

L'ensemble de représentation des documents est un vocabulaire comprenant des termes d'indexation. Ceux-ci sont typiquement les mots les plus significatifs du corpus considéré : noms communs, noms propres, adjectifs... Ils peuvent éventuellement être des constructions plus élaborées comme des expressions ou des entités sémantiques. À chaque élément du vocabulaire est associé un index unique arbitraire. On parle aussi de 'sacs de mots' où les mots sont considérés comme indépendants et où l'ordre est sans importance.

La valeur de chaque caractéristique est appelée le poids du terme et est en général une fonction de fréquence de termes dans le document. Par conséquent, en utilisant la fréquence de

chaque terme comme un poids, les termes qui apparaissent le plus fréquemment sont plus importants et donc descriptifs du document, Documents et requête sont représentés par un vecteur. La représentation d'un document sous forme vectorielle se déroule en deux étapes :

- Extraire les termes pertinents du document.
- Calculer les poids des termes restants.

3.1. Extraction des termes pertinents

Il s'agit de prétraiter le texte des documents textuels en supprimant les mots-vides, la ponctuation et les éventuels 'retours-chariots', de lemmatiser le texte et de le segmenter. Exemple : A partir des documents d1 et d2 donnés dans le figure III .1, après élimination de la ponctuation, élimination des mots-vides et lemmatisation, l'ensemble de termes pertinents pour d1 est : {yeux, si, profond, pench, boir, v, tou, soleil, ven, mir, jet, mour, desesp, perd, memog}; pour d2 : { admir, jour, vaste, parc, pam, sous, œil, brul, soleil, comme, jeune, domin, amour}.figure III.2.

3.2. Calcul des poids

Le poids de chaque terme dans un document peut être obtenu de différentes manières : Booléenne, fréquence des termes, TF-idf (Term frequency - Inverse Document Frequency).

Formellement un modèle vectoriel est défini par un ensemble de vecteurs de base linéairement indépendants.

Chaque contenu est ainsi représenté par un vecteur v, dont la dimension correspond à la taille du vocabulaire. Chaque élément vi du vecteur v consiste en un poids associé au terme d'indice i et à l'échantillon de texte. Un exemple simple est d'identifier vi au nombre d'occurrences du terme i dans l'échantillon de texte. La composante du vecteur représente donc le poids du mot i dans le document. L'un des schémas de pondération les plus usités est le TF-IDF.

3.2.1. Fréquence des termes

Pour la fréquence des termes, le poids d'un terme est obtenu en comptant les occurrences du terme dans le document : tfi; j représente donc la fréquence du terme i dans le document j Exemple : Les représentations vectorielles de d1 et d2 avec la fréquence des termes sont :

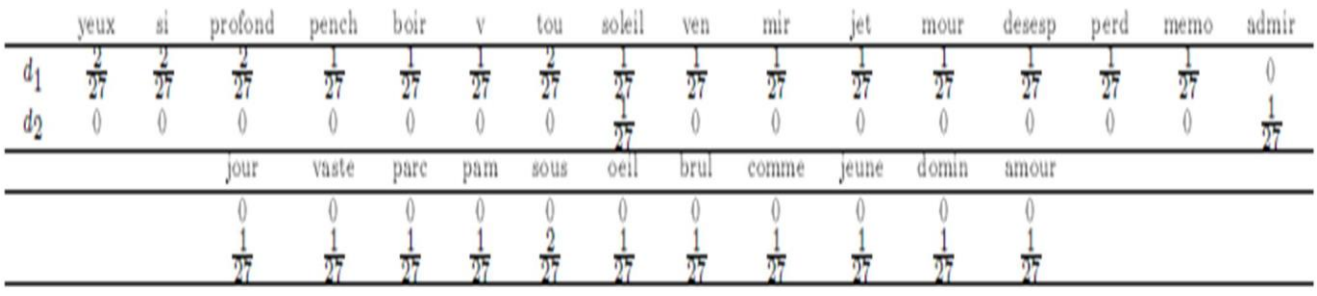

Figure 3.2 : Exemple de fréquence des termes

- La lemmatisation du contenu d'un texte permet de regrouper les mots d'une même famille. Chacun des mots d'un contenu se trouve ainsi réduit en une entité appelée lemme

(forme canonique). La lemmatisation regroupe les différentes formes que peut revêtir un mot, soit : le nom, le pluriel, le verbe à l'infinitif, ...etc.

• Les mots considérés vides, ici, sont : pour, le, la, les, l',..., un, une, des, ..., y, à, se, s', les dérivés du verbe être, les dérivés du verbe avoir, mon, ton, son, mes, tes, ..., qu', que, qui,...

• Quelques exemples de lemmatisation pour cet exemple : yeux est Yeux, profonds est Profond, jeter est jet, désespères est desesp, mémoire est memo, domination est domin, ... L'algorithme de Porter [24] est le plus utilisé.

3.2.2. Contribution d'un terme isolé

• S'il est présent dans le document et la requête, il augmente le score

• S'il est présent dans un des deux seulement, il diminue le score

3.2.3. Tf-Idf

Le TF-idf permet, quant à lui, d'évaluer l'importance d'un terme contenu dans un document, relativement à une collection. Le poids augmente proportionnellement avec le nombre d'occurrences du mot dans le document. Il varie également en fonction de la fréquence du mot dans la collection. Ainsi, la fréquence inverse du document (idf) est une mesure de l'importance du terme dans l'ensemble des documents. Dans le cas du TF-idf , elle vise à donner un poids plus important aux termes les moins fréquents, considérés comme les plus discriminants. Il s'agit de calculer le logarithme de l'inverse de la proportion de documents qui contiennent le terme :

$$idf_i = log \frac{|D|}{|\{d_j : t_i \in d_j\}|} \qquad (1)$$

Où |D| est le nombre total de documents et $|\{d_j : t_i \in d_j\}|$ est le nombre de documents où le terme t_i apparait. Finalement, le poids s'obtient en multipliant les deux mesures

$$tfidf_{i,j} = tf_{i,j}.idf_i \quad (2)$$

Exemple : Notre exemple contient deux documents, donc |D| = 2. Pour le terme "yeux", seul le document d_1 le contient. Par conséquence $idf_{yeux} = log(\frac{2}{1}) \approx 0,301$ De même, le terme "soleil" apparait dans les deux documents, donc

$$idf_{soleil} = log(\frac{2}{2}) = 0.$$

Le calcul du TF-idf du terme "yeux" est donc :

$$tfidf_{yeux,d_1} = tf_{yeux,d_1}.idf_{yeux} = \frac{2}{27}.log\frac{2}{1} \approx 0,0222$$

Pour le document d1 et $tfidf_{yeux,d_2} = tf_{yeux,d_2}.idf_{yeux} = 0.log\frac{2}{1} = 0$

Celui du terme "soleil" est donc :

$$tfidf_{soleil,d_1} = tf_{soleil,d_1}.idf_{soleil} = \frac{1}{27}.log\frac{2}{2} = 0 \text{ pour}$$

Le document d1

$$tfidf_{soleil,d_2} = tf_{soleil,d_2}.idf_{soleil} = \frac{1}{27}.log\frac{2}{2} = 0 \text{ pour le document } d_2.$$

Les représentations vectorielles de d_1 et d_2 avec le TF-idf son :

	yeux	si	profond	pench	boir	v	tou	soleil	ven	mir	jet	mour	desesp	perd	memo	admir
d_1	0,02	0,02	0,02	0,01	0,01	0,01	0,01	0	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0
d_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,01

jour	vaste	parc	pam	sous	oeil	brul	comme	jeune	domin	amour
0	0	0	0	0	0	0	0	0	0	0
0,01	0,01	0,01	0,01	0,02	0,01	0,01	0,01	0,01	0,01	0,01

Figure 3.3 : Les représentations vectorielles de $d_1$et d_2.

4. Proximité entre documents

Étant donnée une représentation vectorielle d'un corpus de documents, on peut introduire une notion d'espace vectoriel sur l'espace des documents en langage naturel. On en arrive à la notion mathématique de proximité entre documents.

En introduisant des mesures de similarité adaptées, on peut quantifier la proximité sémantique entre différents documents. Les mesures de similarité sont choisies en fonction de l'application.

4.1. Mesures de similarité existantes

Une mesure de similarité est en général une fonction qui quantile le rapport entre deux objets, comparés en fonction de leurs points de ressemblance et de dissemblance. Les deux objets comparés sont bien entendu de même type. Toutes les mesures de similarité ne sont pas des métriques.

4.2. Similarité syntaxique

En mathématiques et en informatique, une mesure permettant de comparer des documents textuels, consiste à comparer des chaines de caractères. C'est une métrique qui mesure la similarité ou la dis similarité entre deux chaînes de caractères. Par exemple, les chaînes de caractères "Sam" et "Samuel" peuvent être considérées comme similaires. Une telle mesure sur les chaines de caractères fournit une valeur obtenue algorithmiquement.

Parmi de telles mesures de similarité, citons par exemple, Similarité Cosinus, la distance euclidienne le coefficient de Jaccard, Indice de Dice…etc.

4.3. Similarité Cosinus

La similarité cosinus est fréquemment utilisée [1] en tant que mesure de ressemblance entre deux documents d1 et d2. Il s'agit de calculer le cosinus de l'angle entre les représentations vectorielles des documents à comparer. La similarité obtenue simcosinus(d1; d2) ϵ [0; 1].

$$sim_{cosinus}(d_1, d_2) = \frac{\vec{d_1}.\vec{d_2}}{\|\vec{d_1}\|\|\vec{d_2}\|} \quad (3)$$

Exemple : Par souci de lisibilité et de facilité des calculs, nous nous limitons ici à des vecteurs de taille 6 contenant les termes : {yeux, profond, soleil, memo, sous, œil}.

Les représentations vectorielles de d1, d2 et d3 avec le TF sont :

	yeux	profond	soleil	memo	sous	oeil
d_1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$	0	0
d_2	0	0	$\frac{1}{6}$	0	$\frac{1}{3}$	$\frac{1}{6}$
d_3	$\frac{1}{6}$	0	$\frac{1}{6}$	0	0	0

Nous avons donc simcosinus ϵ (d1; d2)

De la même manière, simcosinus (d1; d3)

$$0,447 \text{ et } sim_{cosinus}(d_2, d_3) \approx 0,288.$$

Par conséquent, selon la similarité cosinus, d1 et d3 sont les plus similaires.

Cette mesure n'est pas sensible à la norme des vecteurs, donc ne tient pas compte de la longueur des documents.

Un avantage de la similarité cosinus est qu'elle peut efficacement profiter d'une implémentation par index inversé à condition d'indexer également la norme des documents. Chaque élément non nul de la requête q permet de retrouver des documents potentiellement pertinents et le produit scalaire (numérateur de la similarité cosinus) est simultanément calculé par accumulation.

Une alternative tout aussi efficace est de calculer le carré de la norme L2 entre q et d_1 exprimée par:

$$\|q - d_1\|_2^2 = \|q\|_2^2 + \|d_1\|_2^2 - 2d_1 \cdot q \quad (4)$$

4.3.1. Distance euclidienne

La distance euclidienne calcule la similarité entre deux documents d1 et d2 comme la distance entre leurs représentations vectorielles ramenées à un seul point.

$$sim_{euclidienne}(d_1, d_2) = \| \vec{d_1} - \vec{d_2} \| = \sqrt{\sum_{i=1}^{n}(d_{1_i} - d_{2_i})^2} \quad (5)$$

Où n est le nombre total de termes représentés. la taille des vecteurs.

Exemple : En gardant des vecteurs de taille 6, nous avons

$$(d_1, d_2) = \| \vec{d_1} - \vec{d_2} \| = \sqrt{\sum_{i=1}^{n}(d_{1_i} - d_{2_i})^2} = \sqrt{\frac{10}{27}} \approx 0,608.$$

De la même manière, distance euclidienne (d1; d3)

$\approx 0,408$ et $sim_{euclidienne}(d_2, d_3) \approx 0,408.$

Par conséquent, selon la distance euclidienne, d3 est à la même distance de d1 que de d2.

4.3.2. Coefficient de Jaccard

L'indice de Jaccard ou coefficient de Jaccard [10] est le rapport entre la cardinalité (la taille) de l'intersection des ensembles considérés et la cardinalité de l'union des ensembles. Il permet d'évaluer la similarité entre les ensembles. Les documents d1 et d2 sont donc représentés, non pas comme des vecteurs, Mais comme des ensembles de termes. La similarité obtenue simjaccard (d1 ; d2) appartient au [0; 1].

$$sim_{jaccard}(d_1, d_2) = \frac{\|d_1 \cap d_2\|}{\|d_1 \cup d_2\|} \quad \textbf{(5)}$$

Il est aussi possible d'utiliser la représentation vectorielle.

$$sim_{jaccard}(d_1, d_2) = \frac{\vec{d_1}.\vec{d_2}}{\|\vec{d_1}\|\|\vec{d_2}\| - \vec{d_1}.\vec{d_2}} \quad (6)$$

Exemple : En gardant des vecteurs de taille 6, nous avons

$$sim_{jaccard}(d_1, d_2) = \frac{\vec{d_1}.\vec{d_2}}{\|\vec{d_1}\|\|\vec{d_2}\| - \vec{d_1}.\vec{d_2}} = \frac{\frac{1}{36}}{\sqrt{\frac{10}{36}} \cdot \sqrt{\frac{1}{6}} - \frac{1}{36}} \approx 0,148.$$

De la même manière

$$sim_{jaccard}(d_1, d_3) \approx 0,809 \text{ et } sim_{jaccard}(d_2, d_3) \approx 0,405.$$

Par conséquent, selon le coefficient de Jaccard, d_1 et d_3 sont les plus similaires.

4.3.3. Indice de Dice

L'indice de Dice mesure la similarité entre deux documents d1 et d2 en se basant sur le nombre de termes communs à d1 et d2

$$sim_{dice}(d_1, d_2) = \frac{2N_c}{N_1 + N_2}$$

Où Nc est le nombre de termes communs à d1 et d2, et N1 (resp. N2) est1 le nombre de termes de d1 (resp. d2).

Exemple : Les documents d1 et d2 ont deux termes en commun, à savoir "se" et "la". Le document d1 contient 36 mots. Le document d2 contient 21 mots. Ainsi, l'indice de Dice entre les documents d1 et d2 vaut : .

$sim_{dice}(d_1, d_2) = \frac{2N_c}{N_1 + N_2} = \frac{2*2}{36+21} \approx 0,07$. De la même manière, $sim_{dice}(d_1, d_3) \approx 0,204$ et $sim_{dice}(d_2, d_3) \approx$

0.117 De la même manière, Par conséquent, selon l'indice de Dice, d1 et d3 sont les plus similaires. [8] [29]

5. Pertinence

Dans un ensemble des documents se trouvant dans un corpus documentaire ou encore sur le web. Le but est de permettre aux utilisateurs de retrouver les documents dont le contenu répond à leur besoin en information, il s'agit donc de retourner l'ensemble de documents pertinents. Cependant, nous constatons que la notion de pertinence dépend de la satisfaction de l'utilisateur d'une part, et des différents sens portés par les termes de la requête d'une autre part. Cette constatation constitue le point faible de la recherche d'information classique, elle représente également le point de départ pour de nouveaux paradigmes de recherche

La notion de pertinence est souvent difficile à appréhender, on parle souvent de deux

Types de pertinence :

5.1. Pertinence utilisateur

Le langage de requête est plus simple les performances sont meilleures grâce à la pondération des termes. Un appariement partiel qui permet de retrouver les documents qui répondent en partie à la requête. Le document est jugé pertinent par l'utilisateur en fonction de son besoin en information

5.2. Pertinence système

Le langage de requête est plus simple les performances sont meilleures grâce à la pondération des termes. Un appariement partiel qui permet de retrouver les documents qui répondent en partie à la requête. [16]

6. Applications

Parmi les applications existantes, on peut citer :

- *La catégorisation* : regrouper automatiquement des documents dans des catégories prédéfinies.

- *La classification* : étant donné un ensemble de documents, déterminer automatiquement les catégories qui permettront de séparer les documents de la meilleure façon possible (catégorisation non supervisée).

- *La recherche documentaire* : trouver les documents qui répondent le mieux à une requête (ce que fait un moteur de recherche) ; la requête de l'utilisateur est considérée comme un document, traduite en vecteur, et comparée aux vecteurs contenus dans le corpus des documents indexés.

- *Le filtrage* : classer à la volée des documents dans des catégories prédéfinies (par exemple, identifier un spam sur la base d'un nombre suspect d'occurrence du mot « pénis » dans un mail et l'envoyer automatiquement à la corbeille).

6.1. Avantages

Quelque soit la technique utilisée, basée sur le modèle vectoriel, a le même format initial à savoir la représentation vectorielle.

Les techniques basées sur le modèle vectoriel sont faciles à développer, il s'agit uniquement de calcul vectoriel.

Le langage de requête est plus simple (liste de mot-clé).

Les performances sont meilleures grâce à la pondération des termes.

Le renvoi de documents à pertinence partielle est possible.

La fonction d'appariement permet de trier les documents.

Le modèle vectoriel est relativement simple à appréhender (algèbre linéaire) et est facile à implémenter. Il permet de retrouver assez efficacement des documents dans un corpus non structuré (recherche d'information), son efficacité dépendant pour une grande part de la qualité de la représentation (vocabulaire et schéma de pondération). La représentation vectorielle permet aussi une mise en correspondance des documents avec une requête imparfaite.

Il comporte également plusieurs limitations qui furent, pour certaines, corrigées par des affinements du modèle. En particulier, ce modèle suppose que les termes représentatifs sont indépendants. Ainsi, dans un texte, l'ordre des mots n'est pas pris en compte. Dans sa version la plus simple, il ne prend pas non plus en compte les synonymes ou la morphologie des contenus.

Simple et efficace à utiliser dans des systèmes de recherche d’information. [23]

6.2. Inconvénients

Des mots identiques considérés comme peu pertinents peuvent parfois trop influer sur la valeur de la similarité. Par exemple, pour les phrases "Tout est bien qui finit bien" et "C'est notre seul bien", le terme "est" n'est pas vraiment pertinent et pourtant, il va avoir un poids certain.

Notons cependant que la lemmatisation, l'élimination des mots-vides et le TF-idf permettent de pallier cet inconvénient. [29]

7. Conclusion

Dans ce second chapitre, nous sommes essentiellement intéressés à l'étude du modèle vectoriel. On a commencé par le principe du modèle vectoriel dans la RI dent a parlé de l'interprétation de ce modèle, de TF idf et de similarité entre les documents et la pertinence.

Alors il contribue à la résolution des problèmes inhérents à la recherche d'information et il récupère tous les documents dont il a besoin d'une façon rapide et efficace.

CHAPITRE 3

Conception et implémentation

1. Introduction :

Afin de réduire la complexité des documents et les rendre plus faciles à manipuler, le document doit être transformé. Dans notre SRI, il est nécessaire de pouvoir rechercher les documents de la collection dont le contenu correspond au contenu de la requête. La recherche implique une méthode de tri et la comparaison de contenu implique une analyse à défaut de pouvoir directement comparer les concepts véhiculés dans le document à ceux présents dans la requête.

I.7. Notre système de recherche d'informations (SRI) :

1.1. Processus de RI proposé :

L'objectif principal de notre travail est d'estimer la pertinence des documents par rapport à une requête donnée de manière à ce que le système puisse ordonner les documents en fonction de leur degré de pertinence qui est la mesure du cosinus.

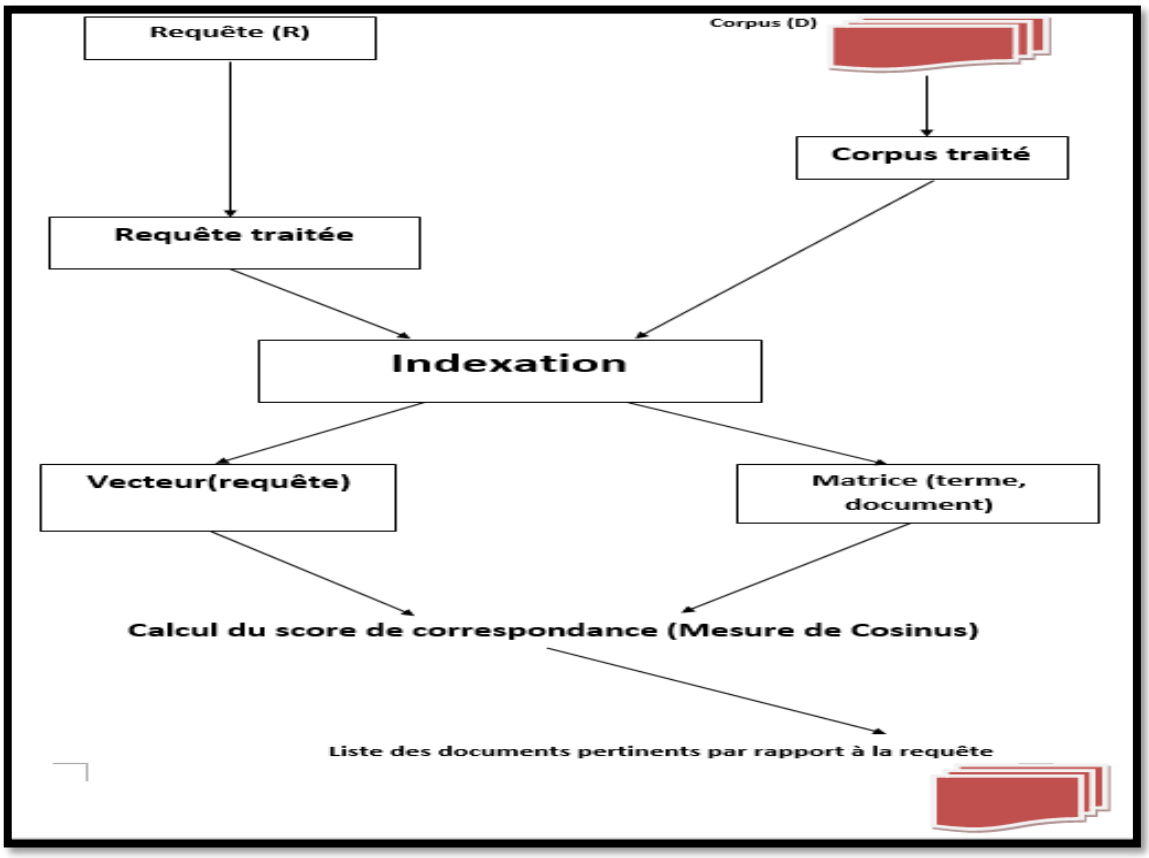

Figure 4.1 : Schéma explicatif du processus proposé

Ce processus de recherche peut se résumer dans la figure (figure) ci-dessus :

- L'indexation des documents et des requêtes.
- L'appariement (D , R), qui permet de comparer la requête et le document.

L’utilisateur exprime sa requête (son Besoin en information) et interroge le corpus des documents via une interface qui assure la communication entre la base documentaire et l’utilisateur. Notre SRI est constitué de module de traitement des documents pour l’indexation et le stockage d’information, module de traitement des requêtes qui à pour objectif de représenter les requêtes des utilisateurs et module de recherche d'information en effectuant une correspondance entre requête utilisateur et documents de la base, le système renvoi les documents pertinents en fin de processus de recherche.

Avant que le processus de recherche puisse être lancé, il est important de définir la base de données.

1.1.1. Indexation :

L'indexation est l'une des phases principales de notre processus de recherche d'informations. Indexer un document c’est élire ses termes représentatifs afin de générer la liste des termes d’indexation et ajouter à l’index de la collection, pour chacun de ces termes, la liste des références de chaque document le contenant. Par référence on entend identifiant, c'est-à-dire un moyen de retrouver de façon non ambigüe des documents ou un document ou une partie de document où le terme apparaît.

L'indexation consiste donc à associer à chaque document une liste de mots clés appelée aussi descripteur, susceptible de représenter au mieux le contenu des documents.

Les prétraitements appliqués pour chaque document sont :

- L’analyse lexicale :

Elle permet de convertir un texte de document en une liste de termes. Un terme est un groupe de caractères constituant un mot significatif. L’analyse lexicale permet de reconnaître les espaces de séparation des mots, les chiffres, les ponctuations… etc.

- Le Nettoyage :

Afin de ne garder que les termes importants, nous utilisons un anti-dictionnaire (stoplist). C’est une liste de mots vides qui contient généralement les séparateurs, les articles, pronoms, prépositions, les mots outils, ainsi que les mots athématiques c'est-à-dire présents dans le document pour l’introduire ou le présenter mais n’ayant pas de réels rapports avec le sujet traité. Cette liste et la base documentaire doivent être de la même langue. Pour notre SRI, nous avons utilisé un tableau pour les stopwords en anglais, français et arabe.

Le traitement lié à un anti-dictionnaire (figure 4.11) est très simple. Quand un mot est rencontré dans un texte à indexer, s'il apparaît dans l’anti-dictionnaire, il n’est pas considéré comme un index.

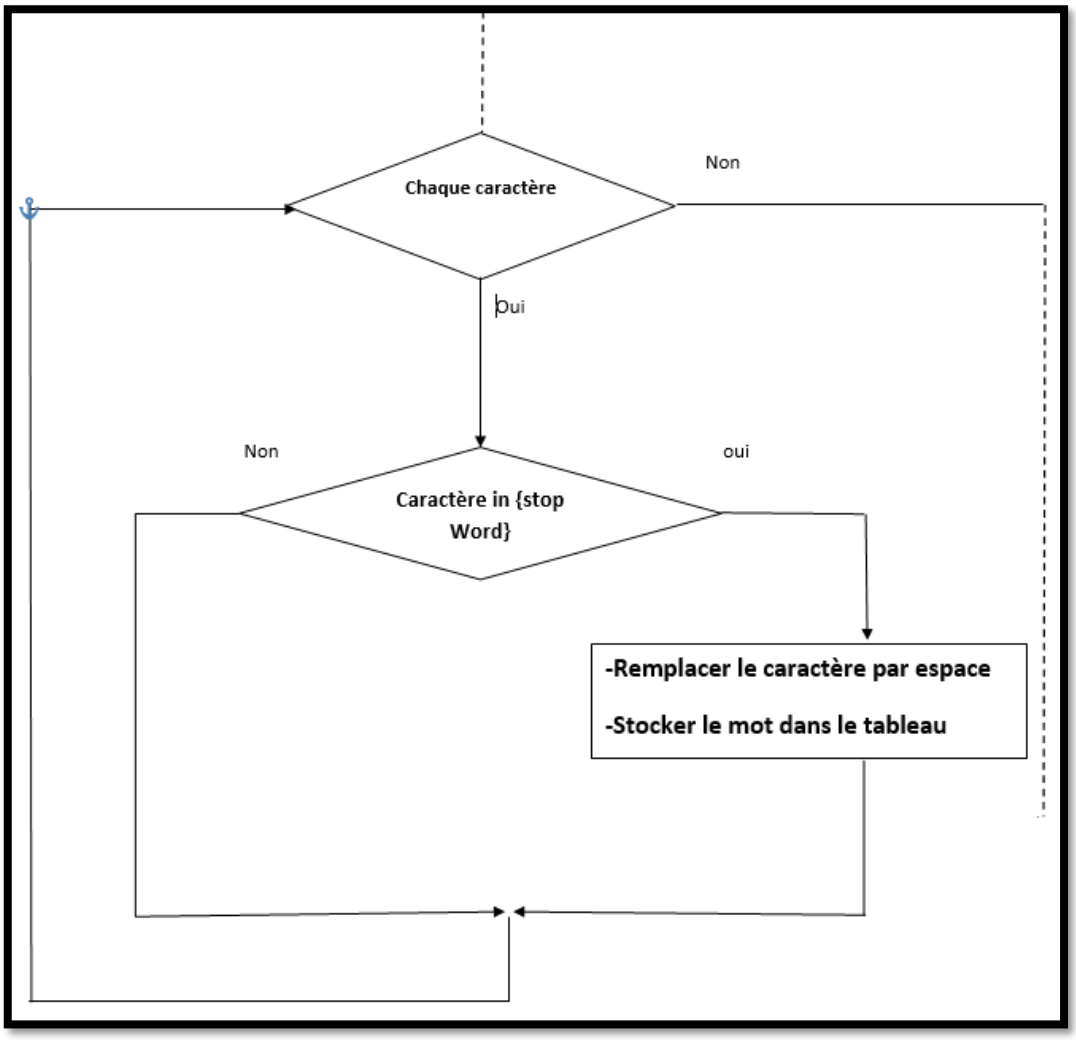

Figure 4.2 : traitement de nettoyage

Ce traitement de nettoyage permet de réduire la taille du tableau.

A la fin de cette étape, chaque document sera représenté par un vecteur de mots (sans séparateurs), la figure 4.12 illustre un exemple en éliminant les séparateurs.

La figure 4.11 présente les caractères séparateurs utilisés dans notre système :

```
var stopWords = new string[] { "1","2","3","4","5","6","7","8","9","0",".","{","/","]",";",":","-","+","*","}",")","(","[",">","<","%",",",".","?",
    " avec "," avoir "," bon "," ça "," car "," ce "," ceci "," cela "," celà "," ces "," c'est "," cet "," cette "," ceux "," chaque "," ci ",
    " comme "," comment "," dans "," debut "," début "," dedans "," dehors "," depuis "," des "," deux "," devrait "," doit "," donc ",
    " dos "," droite "," du "," d'un "," d'une "," egalement "," elle "," elles "," en "," encore "," ensuite "," es "," essai "," est ",
    " et "," étaient "," étais "," était "," étant "," état "," etc "," ete "," été "," étée "," étées "," êtes "," étés ",
    " étiez "," étions "," être "," eu "," eux "," fait "," faites "," fois "," font "," force "," fûmes "," furent "," fus "," fusse ",
    " fussent"," fusses"," fussiez "," fussions "," fut "," fût "," fûtes "," grace "," grâce "," haut "," hors "," ici "," il "," ils ",
    " ils4les "," je "," juste "," la "," Le "," là "," le "," La "," les "," Les "," leur "," leurs "," ma "," maintenant "," mais "," meme "," mêm
    " mieux "," mine "," moins "," mon "," mot "," ni "," nommés "," nos "," notre "," nous"," nouveaux "," ou "," où "," par ",
    " parce"," parole "," pas "," personnes "," peu "," peut "," pièce "," plupart "," pour "," pourquoi "," quand "," que ","quel ",
    " quelle"," quelles "," quels "," qui "," sa "," sans "," sera "," serai "," seraient "," serais "," serait "," seras "," serez ",
    " seriez "," serions "," serons "," seront "," ses "," seulement "," si "," sien "," soi "," soient "," sois "," soit "," sommes ",
    " son"," sont "," sous "," soyez "," soyons "," suis"," sujet "," sur "," ta "," tandis ","tellement "," tels "," tes "," ton ",
    " tous "," tout "," toute "," toutes "," tres "," très "," trop "," tu "," un "," une "," valeur "," voie "," voient "," vont ",
    " vos "," votre "," vous "," vu ",
   "الت","وكان","وقال","هناك","مقابل","لقاء","ولا","وقد","منذ","لها","كما","قوة","هي","هو","من","له","لن","لم","كل","في",".",
   "اما ","الا","ا","حيث","يمكن","يكون","منها","فيها","يوم","وهي","وهو","ومن","ولم","وفي","لكن","فيه","وكانت ",
   "ف","انها","اول","انه","الى","حين","حول","دون","ذلك","ابين","الذين","الان","الذي","الذى","ايضا","اكثر","التي","التى","ا
   "وان","هذه","نحو","لدى","كان","قال","قبل","فان","واضافت","واضاف","واحد","هذا","مع","ما","لا","قد"," و ",
   "بان","احد","اذا","حتى","بعض","بعد","ضد","تم","عليها","عليه","على","عندما","عند","عن","،",",","أ","ا","ب","واوضح","كانت",
   "بن","به","ثم","اف","ان","او","اي","بها "


};
     a[i] = a[i].ToLower();
    for (int n = 0; n < stopWords.Length; n++)
    {
        a[i] = a[i].Replace(stopWords[n], "\n");

    }
```

Figure 4.3 : Anti-dictionnaire (les stops Word)

Texte	Suppression les stops Word
Txt1	Diagramme de cas d’utilisation représente la structure des fonctionnalités nécessaires aux utilisateurs du système
Txt1 sans Stop Words	Diagramme cas d’utilisation représente structure fonctionnalités nécessaires utilisateurs système

Figure 4.4 : Exemple sur la suppression de séparateurs

- Elimination des majuscules :

Cette étape rend tout caractère lu en minuscule.

- Lemmatisation des mots :

Beaucoup de mots ont des formes différentes, mais leur sens reste le même ou très similaire et notamment dans le cas des mots conjugués. Par exemple : les mots « transmettra » et « transmet » sont considérés comme des descripteurs différents alors qu’il s’agit de deux formes conjuguées du même verbe qui ont le même sens. De ce fait, la technique de lemmatisation consiste à remplacer chaque mot par son lemme. Elle nécessite une analyse grammaticale des textes afin de remplacer les verbes par leur forme infinitive et les noms par leur forme au singulier.

Plusieurs algorithmes ont été proposés, le plus célèbre pour la langue anglaise utilisé est « l’algorithme de Porter » [ ], ce dernier permet de supprimer les affixes des mots (suffixes, préfixes, post-fixe) afin d’obtenir une forme canonique du mot.

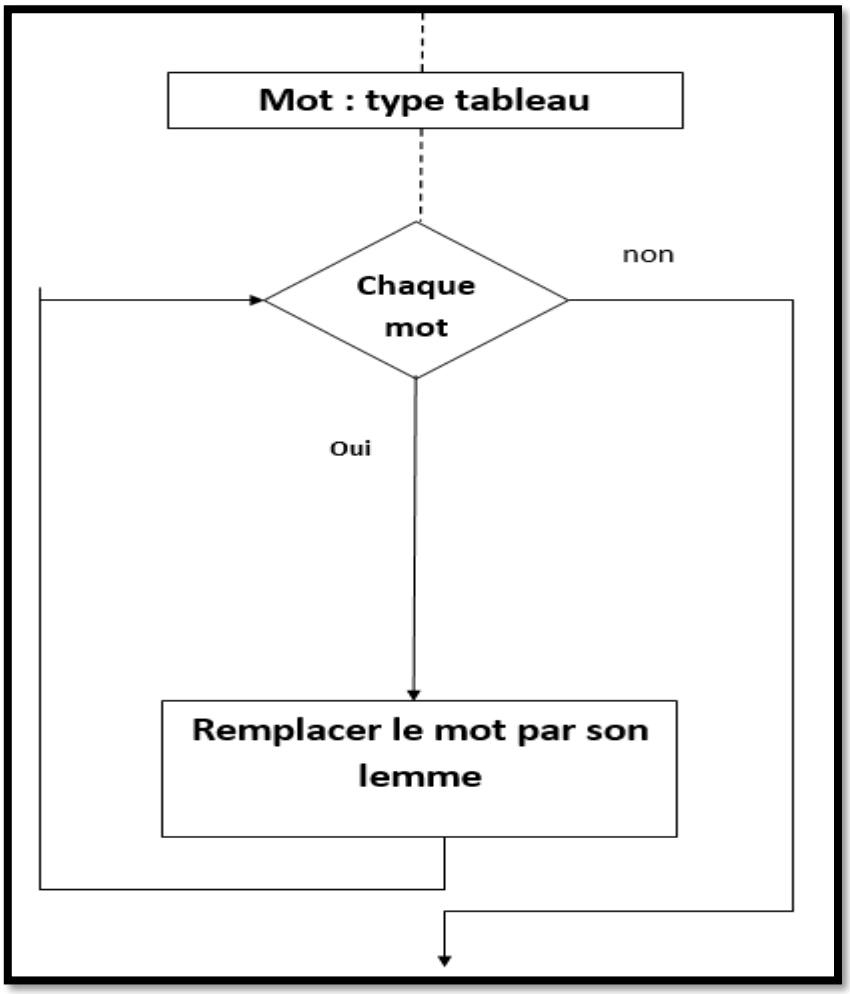

Figure 4.5: Lemmatisation des mots

Cette étape permet de réduire le nombre de termes dans le tableau et permet de représenter par un même descripteur des mots qui ont le même sens.

Après ces prétraitements, il est nécessaire de définir un tableau de mots qui contient tous les mots existant dans le corpus et une matrice (tableau à deux dimensions) dont les lignes représentent les documents et les colonnes représentent les mots.

Après ces étapes nous construisons une structure d'index pour représenter les documents ainsi l'ensemble des mots qui représentent le mieux le contenu d'un document. A partir de cette liste de mots appelés aussi descripteurs, on calcule leurs degrés d'importance dans les documents, c'est la notion de pondération.

1.1.2. La pondération :

La pondération d'un terme d'indexation est l'association de valeurs numériques à ce terme de manière à représenter son pouvoir de discrimination pour chaque document de la collection. Cette caractérisation est liée au pouvoir informatif du terme pour le document donné en calculant leur poids.

La pondération *Tf*idf* (formule) ne prend pas en considération le fait que les documents peuvent être de tailles différentes . Si par exemple deux mots ont le même poids de *Tf*idf*, ils n'ont pas forcément la même importance, car il se peut que l'un des deux se trouve dans un document de longueur plus grande que l'autre alors ce dernier sera moins important.

La formule $tf \times idf$ fournit une bonne représentation du poids pour les corpus dont les documents sont de taille homogène c'est-à-dire composés de documents de tailles similaires.

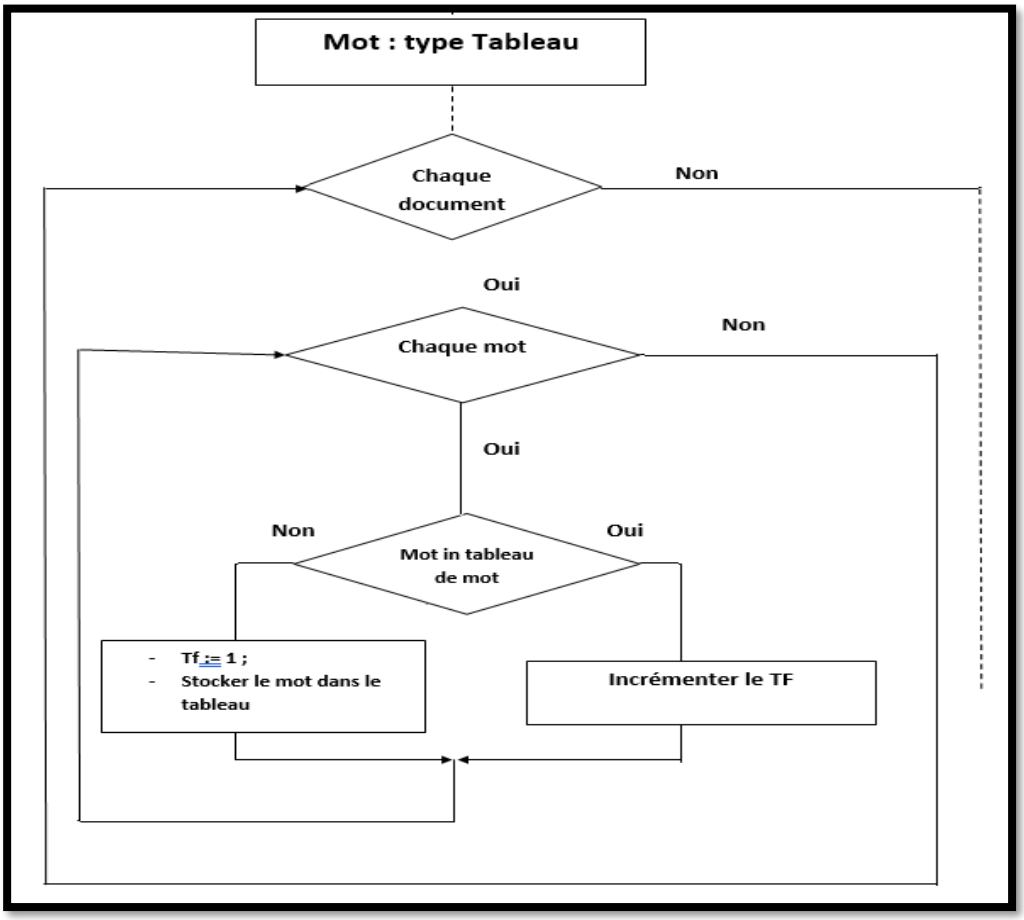

Figure 4.6: Calcul du tf

La requête de l'utilisateur va être indexée de la même façon que les documents. Mais la pondération de la requête consiste à affecter le poids '1'au mot de la requête s'il existe dans le tableau des mots, sinon on lui affecte le poids '0'.

1.1.3. L'appariement document-requête :

L'appariement entre le document et la requête revient à calculer un score représentatif de la ressemblance entre le document et la requête. Dans le modèle vectoriel, ce score consiste à trouver les vecteurs documents qui s'approchent le plus de vecteur de la requête. Cette mesure de similarité est obtenu par l'évaluation de la mesure du cosinus (formule).

La mesure du cosinus mesure l'angle formé entre le vecteur document et le vecteur requête, plus ces vecteurs sont proches, plus l'angle formé est petit, et plus le cosinus de cet angle est grand. En fait, l'avantage de cette mesure est qu'elle est normalisée, le cosinus vaut '1' si les vecteurs sont confondus et '0' s'ils sont orthogonaux, alors on aura des valeurs comprises entre '0' et '1' pour chaque document du corpus, telle que la bonne valeur sa sera la plus proche au 1 et donc c'est le document le plus pertinent.

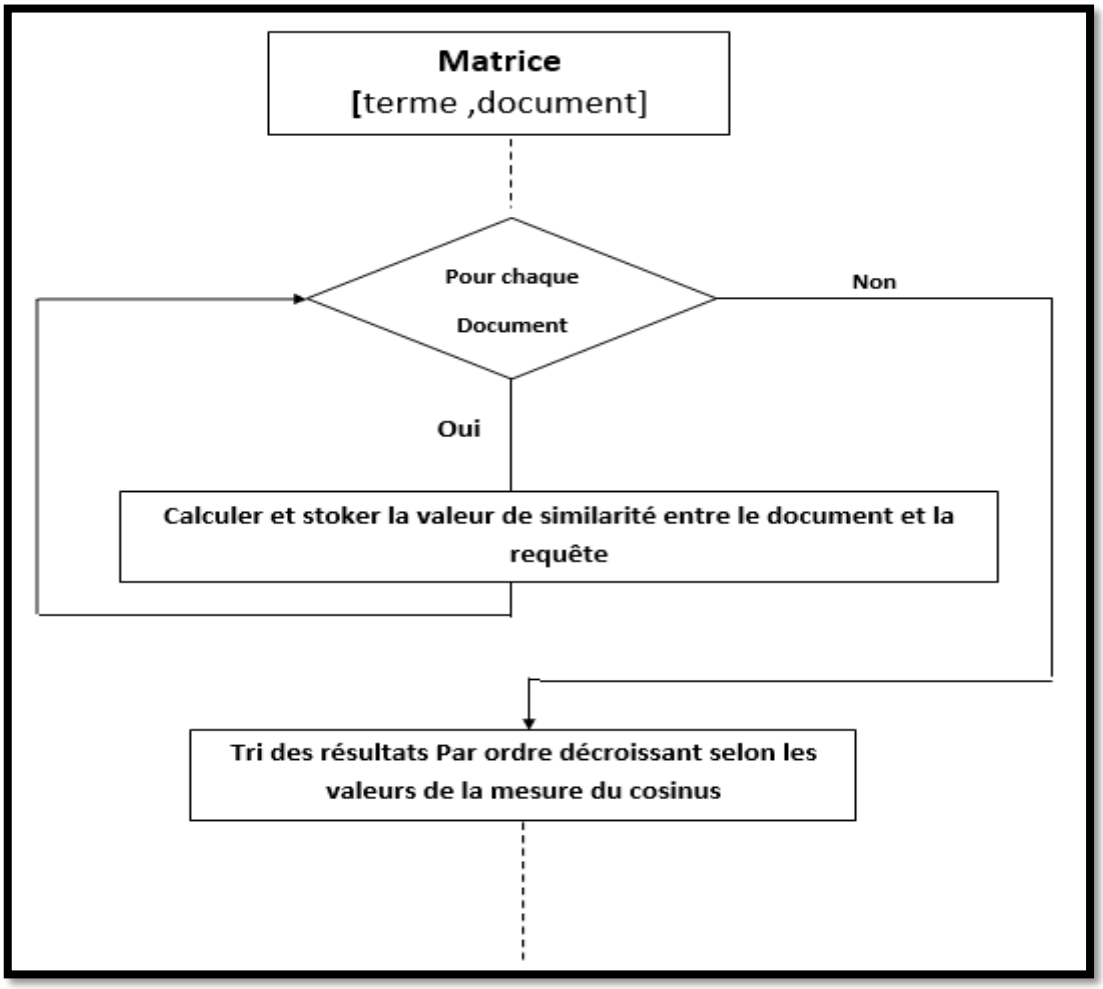

Figure 4.7 : Appariement document-requête

Le processus d'indexation et la fonction d'appariement constituent les deux éléments essentiels du modèle de recherche.

2. Présentation des outils utilisés :

La conception et l'implémentation de notre SRI repose sur les outils suivants :

2.1. UML :

UML n'est pas une méthode, en fait, et comme son nom l'indique est un langage graphique de modélisations des données et des traitements, fonde sur des concepts orientes objets. UML2.0

comporte treize types de diagrammes représentant des vues distinctes du système d'information. Ils se répartissent en trois grands groupes.

- Diagrammes structurels ou diagrammes statiques
- Diagrammes comportement ou diagrammes dynamiques
- Diagrammes d'interaction

Ces diagrammes, d'une utilité variable selon les cas, ne sont pas nécessairement tous produits à chaque modélisation.

2.2. ASP.NET :

Est un Framework permettant de générer à la demande des pages web, lancée par Microsoft en juillet 2000, et utilisée pour mettre en œuvre des applications web . Il s'agit d'une évolution majeure d'Active Server Pages (ASP, alias Classique ASP), par laquelle cette technique a été incorporée dans la plateforme Microsoft .NET. Le moteur d'ASP.NET est un filtre branché sur le serveur web Internet Information Services (IIS). Il est distribué avec le Framework .NET. ASP.NET peut être utilisé avec n'importe quel langage de programmation pour la plateforme .NET (Visual Basic .NET, C#, JScript …) [ ].

2.3. Editeur de programmation Visual studio basic

Microsoft Visual Studio est une suite de logiciels de développement pour Windows et mac OS conçue par Microsoft.

La dernière version s'appelle Visual Studio 2019.

Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications web ASP.NET, des services web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++, Visual C# utilisent tous le même environnement de développement intégré (IDE), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités du Framework .NET, qui fournit un accès à des technologies clés simplifiant le développement d'applications web ASP et de services web XML grâce à Visual Web Développer. Durant sa conférence Connecté ( ) 2016, Microsoft a annoncé le portage de Visual Studio sur mac OS, le système d'exploitation d'Apple.

Langages supportés par Visual Studio

_ C++/C
_ C++/CLI
_ C#

_ F#

_ Visual Basic

_ Python

_ Q#

3. Présentation des diagrammes de comportement de notre SRI :

3.1. Diagramme de cas d'utilisation « Interactions client_SRI » :

Les cas d'utilisations décrivent le comportement du système étudié du point de vue de l'utilisateur, et décrivent les possibilités d'interactions fonctionnelles entre le système et les acteurs, ils permettent de définir les limites et les relations entre le système et son environnement. Il est destiné à structurer les besoins des utilisateurs et les objectifs par rapport au système. C'est donc l'image d'une Fonctionnalité en réponse à la simulation d'un acteur externe. Le diagramme de cas d'utilisation ci-dessous décrit les interactions client SRI.

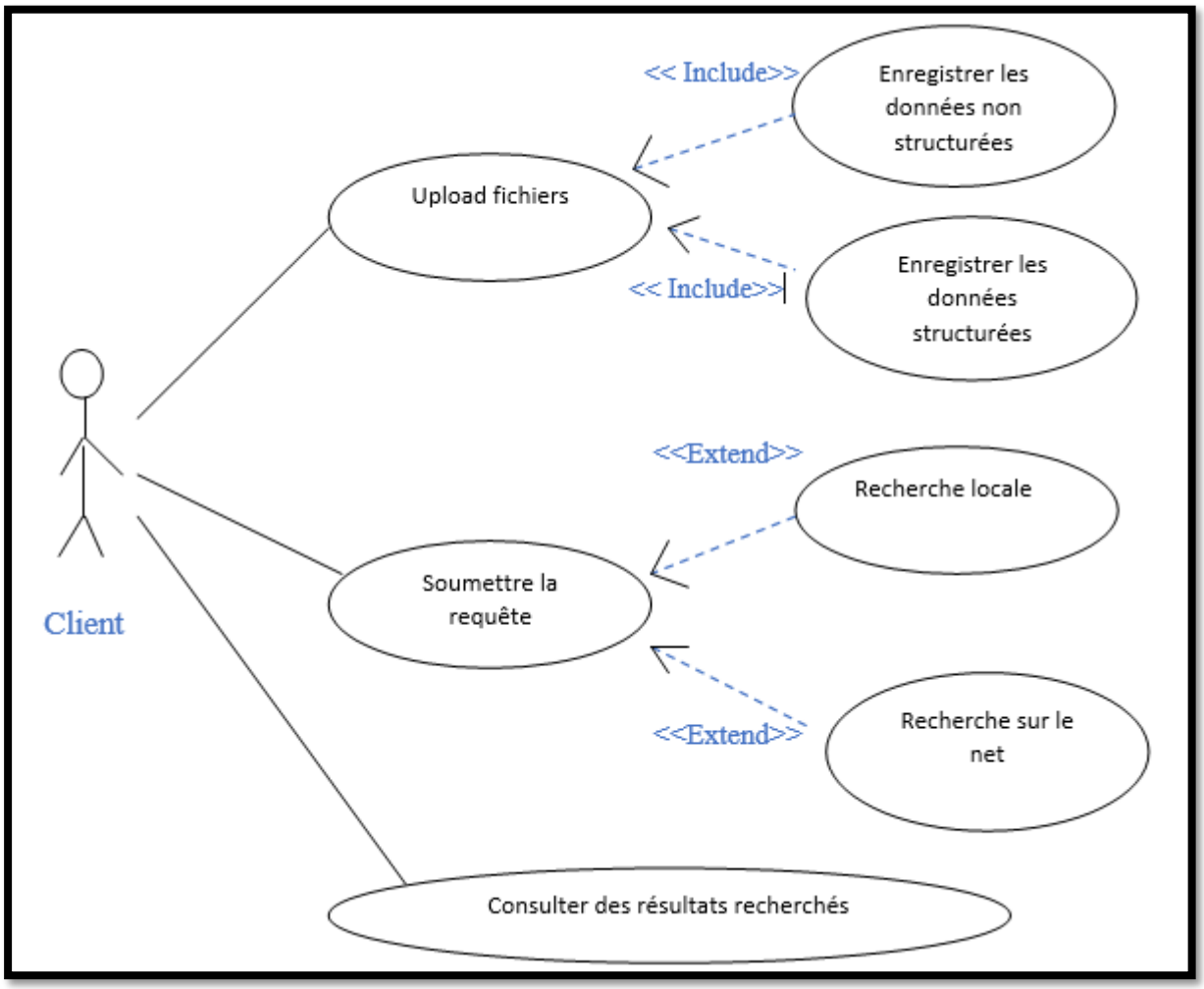

Figure 4.8 : Diagramme cas d'utilisation « Interactions client_SRI »

3.2. Diagrammes de séquences

Les trois diagrammes de séquences présentés ci-dessous (Figure 4.4 ,4.5 et 4.6) permettent de représenter les interactions entre nos différents objets selon un point de vue temporel. L'accent est mis sur la chronologie des envois de messages.

- L'ordre d'envoi d'un message est déterminé par sa position sur l'axe vertical du diagramme et le temps s'écoule '' de haut en bas '' de cet axe.

- La disposition de nos objets sur l'axe horizontal n'a pas de conséquence pour la sémantique du diagramme.

Les différentes périodes d'activité d'un objet sont représentées de manière explicite à la moyenne d'une bande rectangulaire superposé à la ligne de vie de l'objet.

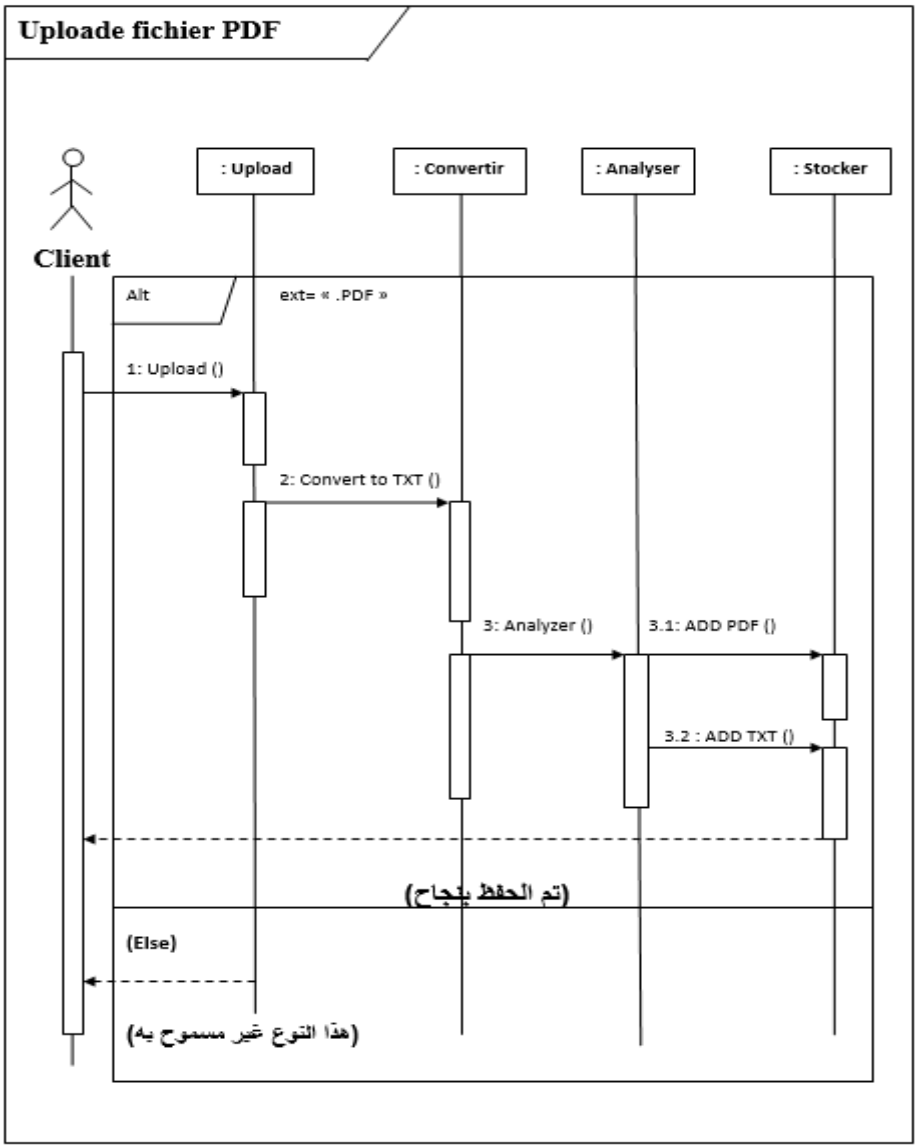

Figure 4.9 : Diagramme de séquence Upload fichier PDF

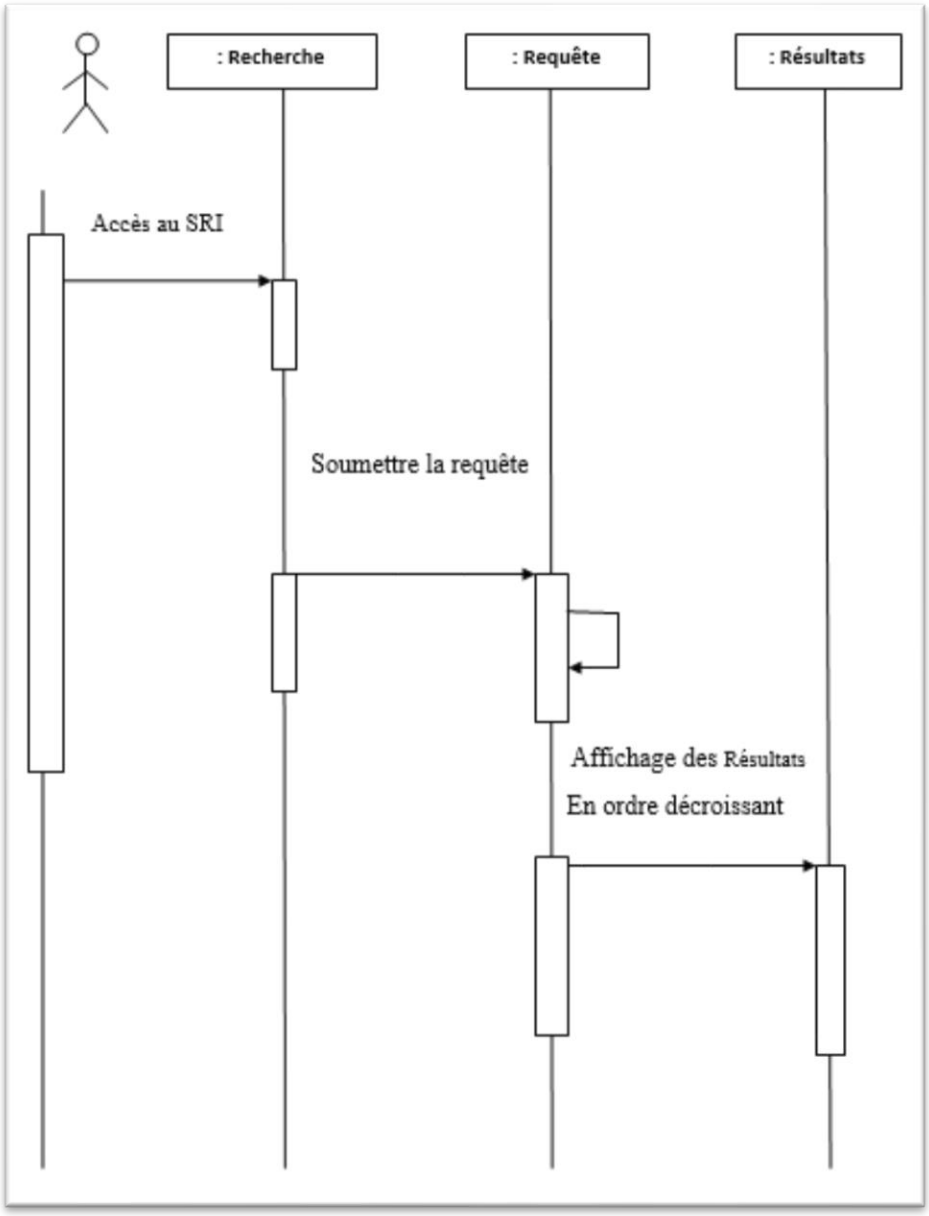

Figure 4.10: Diagramme de séquence « recherche locale »

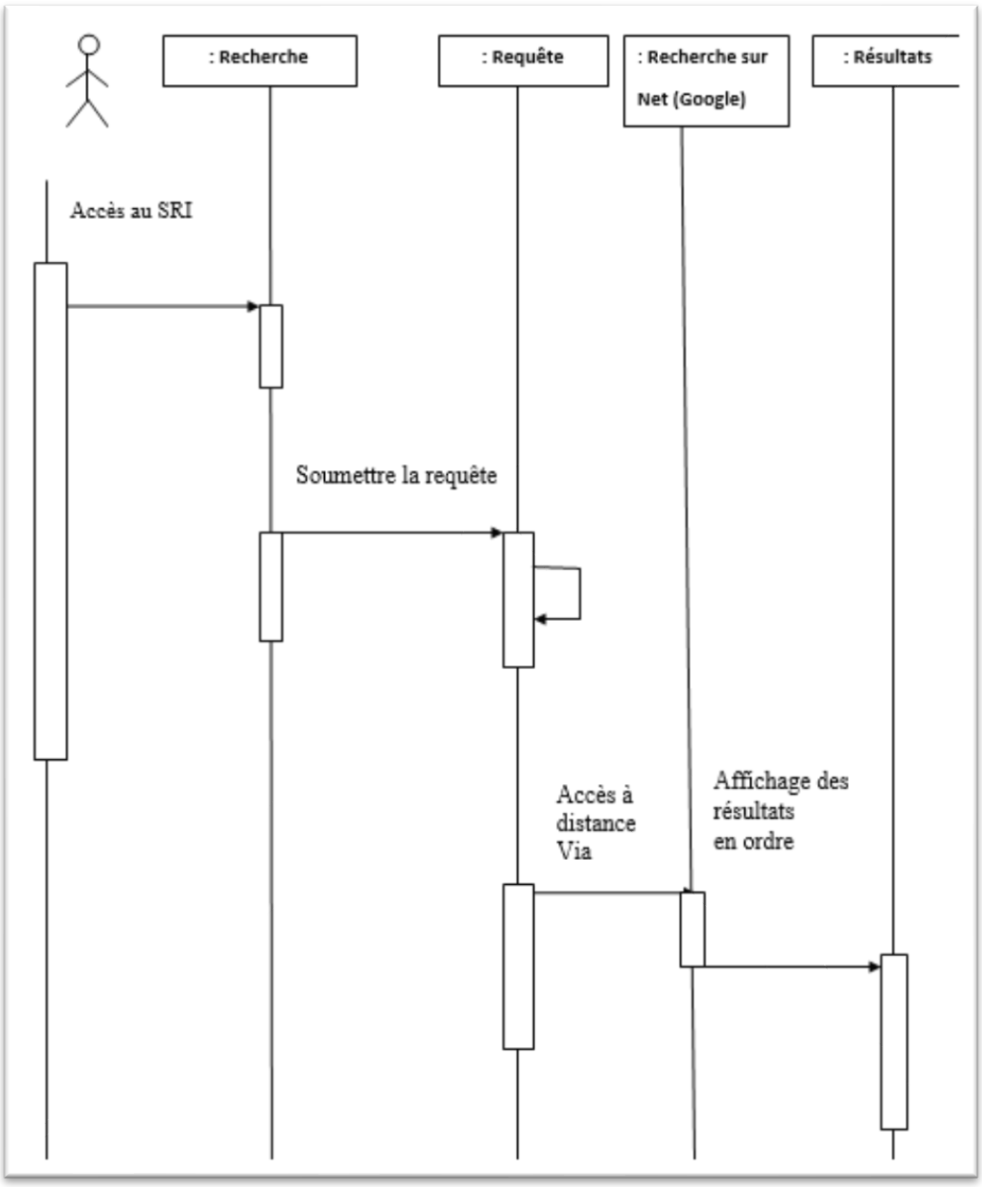

Figure 4.11 : Diagramme de séquence « recherche sur net »

4. Quelques portions de codes sources de notre (SRI) :

A. Choisir et Stocker les fichiers PDF dans un répertoire :

```
if (File.HasFiles)
{
    String ext = System.IO.Path.GetExtension(File.FileName);
    if (ext != ".pdf"  )
    {
        t2.Text = "هذا النوع غير مسموح به ";
        return;
    }
```

Figure 4.12: Code de téléchargement PDF en ASP.NET

B. Enregistrer les caractéristiques et le contenu des fichiers dans la base de données :

```
tring file_name = "";

ile_name = file_name + DateTime.Now.Year.ToString() + DateTime.Now.Month.ToString();
ile_name += DateTime.Now.Hour.ToString();
ile_name += DateTime.Now.Minute.ToString();
ile_name += DateTime.Now.Second.ToString();
ile.SaveAs(Server.MapPath("files") + "\\" + file_name + ext);
tring date;
ate = DateTime.Now.Month.ToString() + "/" + DateTime.Now.Day.ToString() + "/" + DateTime.Now.Year.ToString() + "  " + DateTime.Now.Hour.ToString() +
```

Figure 4.13: Enregistrer les caractéristiques et le contenu des fichiers dans **la base de données**

C. Convertir Doc.pdf vers Doc.txt et les sauvegarder dans le même répertoire :

```
tring path = "~\\files\\" + file_name + ext;

autinSoft.PdfFocus f = new SautinSoft.PdfFocus();


.OpenPdf(Server.MapPath("files") + "\\" + file_name + ext);
.ToText(Server.MapPath("files") + "\\" + file_name + ext + ".txt");
tring sql = "INSERT INTO table_3 ( title, Date1, path, dawl) VALUES ('" + System.IO.Path.GetFileName(File.FileName) + "', '" + date + "', '" + path +
```

Figure 4.14: Convertir Doc.pdf vers Doc.txt et les sauvegarder dans le même répertoire

a. Calculée TF et idf et TF*idf

Calcul du TF

```
private static double getTF(string document, string term)
{
    string[] queryTerms = Regex.Split(document, "\\s");
    double count = 0;


    foreach (string t in queryTerms)
    {
        if (t == term)
        {
            count++;
        }
    }
    return count;

}
```

Figure 4.15 : Calcul du TF

Calcul de l'idf

```
private static double getIDF(string term)
{
    double df = 0.0;
    //get term frequency of all of the sentences except for the query
    for (int i = 1; i < docs.Length; i++)
    {
        if (docs[i].Contains(term))
        {
            df++;
        }
    }

    //Get sentence count
    double D = docs.Length - 1; //excluding the query

    double IDF = 0.0;

    if (df > 0)
    {
        IDF = Math.Log(D / df);
    }

    return IDF;
}
```

Figure 4.16 : Calcul de l'idf

Calcul TF*idf

```
public static double dotproduct(double[] v1, double[] v2)
{
    double product = 0.0;
    if (v1.Length == v2.Length)
    {
        for (int i = 0; i < v1.Length; i++)
        {
            product += v1[i] * v2[i];
        }
    }
    return product;
}
```

Figure 4.17: Calcul TF*idf

b. Calcul de la similarité cosinus

```
public static double cosinetheta(double[] v1, double[] v2)
{
    double lengthV1 = vectorlength(v1);
    double lengthV2 = vectorlength(v2);

    for (int ii = 0; ii < v1.Length; ii++)
    {
        Console.WriteLine(v2[ii] + "\t" + v1[ii]);

    }

    double dotprod = dotproduct(v1, v2);

    return dotprod / (lengthV1 * lengthV2);

}
```

Figure 4.18 : Calcul de la similarité cosinus

c. Affichage des résultats

```
<asp:GridView ID="GridView1" runat="server" Height="240px" Width="926px" AllowPaging="True" AutoGenerateColumns="False" BackColo
    <Columns>
        <asp:BoundField DataField="ID" HeaderText="رقم" InsertVisible="False" ReadOnly="True" SortExpression="ID" />
        <asp:BoundField DataField="TITLE" HeaderText="عنوان " SortExpression="TITLE" />
        <asp:BoundField DataField="DATE1" HeaderText="تاريخ التحميل" SortExpression="DATE1" />
        <asp:BoundField DataField="dawl" SortExpression="dawl" />
        <asp:HyperLinkField DataNavigateUrlFields="path" Text="فتح" />
    </Columns>
    <FooterStyle BackColor="#F7DFB5" ForeColor="#8C4510" />
    <HeaderStyle BackColor="#A55129" Font-Bold="True" ForeColor="White" />
    <PagerStyle ForeColor="#8C4510" HorizontalAlign="Center" />
    <RowStyle BackColor="#FFF7E7" ForeColor="#8C4510" />
    <SelectedRowStyle BackColor="#738A9C" Font-Bold="True" ForeColor="White" />
    <SortedAscendingCellStyle BackColor="#FFF1D4" />
    <SortedAscendingHeaderStyle BackColor="#B95C30" />
    <SortedDescendingCellStyle BackColor="#F1E5CE" />
    <SortedDescendingHeaderStyle BackColor="#93451F" />
</asp:GridView>
</div>
```

Figure 4.19 : Affichage des résultats

4.1.1. Recherche sur le Net

```
protected void Button2_Click(object sender, EventArgs e)
{
    string s = string.Format("http://www.google.co.in/search?q={0}", TextBox1.Text+".pdf");
    Response.Redirect(s);
}
```

Figure 4.20 : La recherche sur le Net

5. Implémentation de l'application

Nous décrivons dans cette section les principales réalisations de notre SRI :

5.1. Page d'accueil :

La figure 4.3 présente la page d'accueil (home page) de notre site web SRI :

Figure 4.21 : Page d'accueil « home page ».

5.1.1. Page uploade-file :

Le bouton « choisir un fichier» amène l'utilisateur à la page de téléchargement des documents :

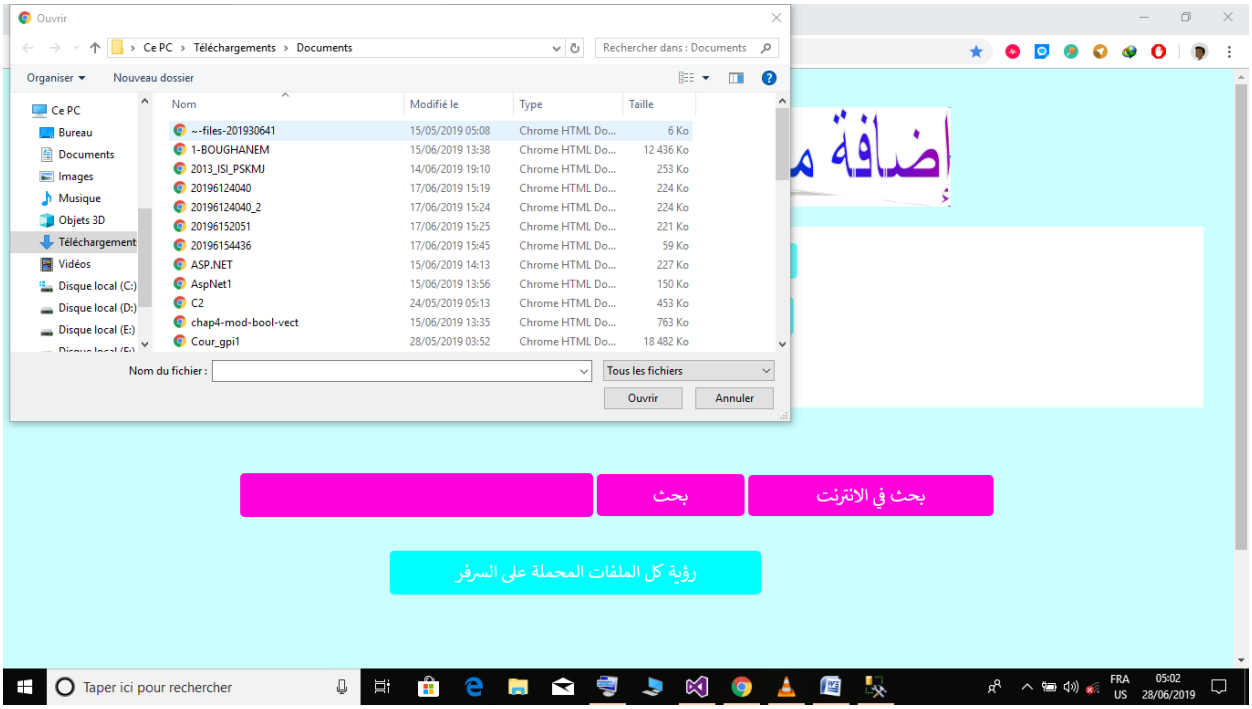

Figure 4.22 : Interface uploade file

5.1.2. le bouton (بحث) :

En cliquant sur le bouton (بحث), l'user peut chercher des documents relatifs dans le corpus.

Figure 4.23 : le bouton (بحث),

Une fois la recherche se termine avec succès, les résultats pertinents apparaissent en ordre décroissant sur l'écran suivant :

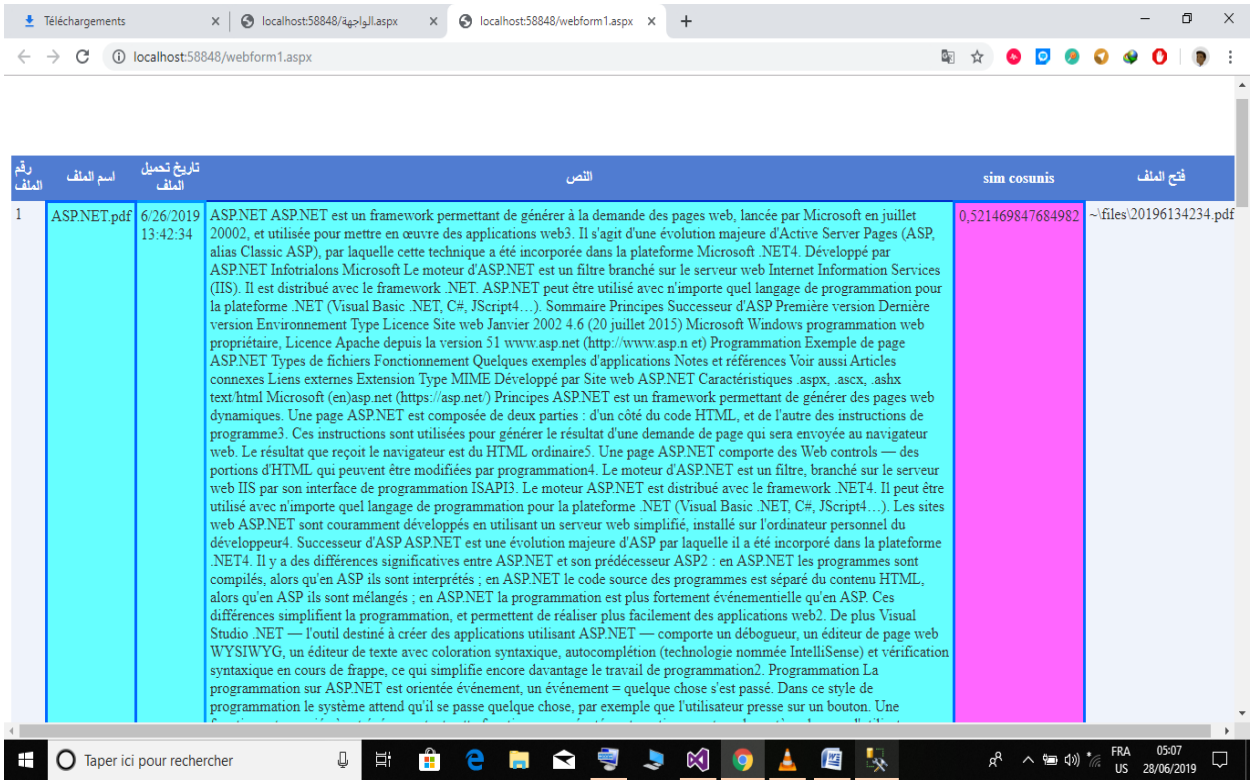

Figure 4.24 : Affichage des résultats pertinents

5.1.3. Bouton de recherche sur net (بحث في الانترنت) :

Ce bouton sert à chercher des résultats pertinents sur web via Google (Figure 4.26).

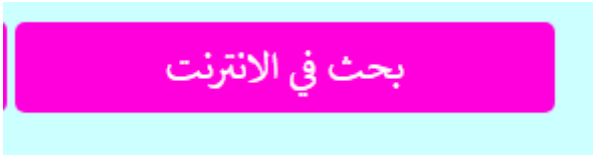

Figure 4.25 : Bouton de recherche sur net

Les résultats restitués, à travers l'écran ci-dessous, sont classés par ordre décroissant en fonction de la mesure de similarité thématique (pertinence thématique) requête-document préprogrammé pour le moteur de recherche google.

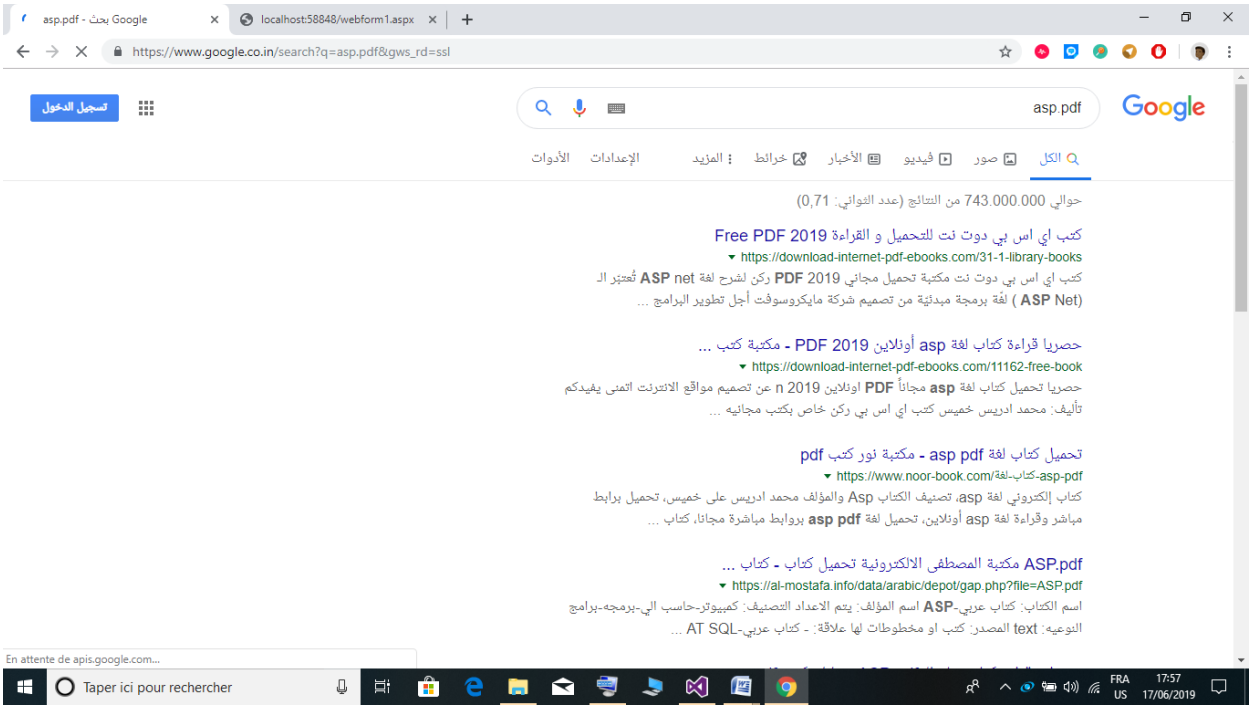

Figure 4.26 : Affichage des résultats

5.1.4. Bouton (رؤية كل الملفات المحملة على السر فر) :

Une fois le bouton enfoncé, toute la collection des documents s'affiche sur écran.

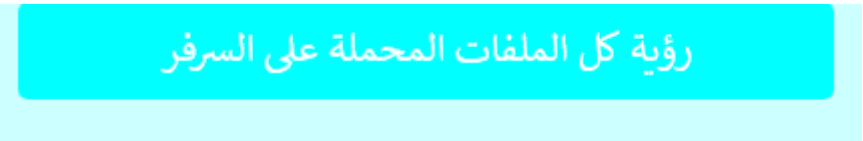

Figure 4.27 : Affichage du corpus

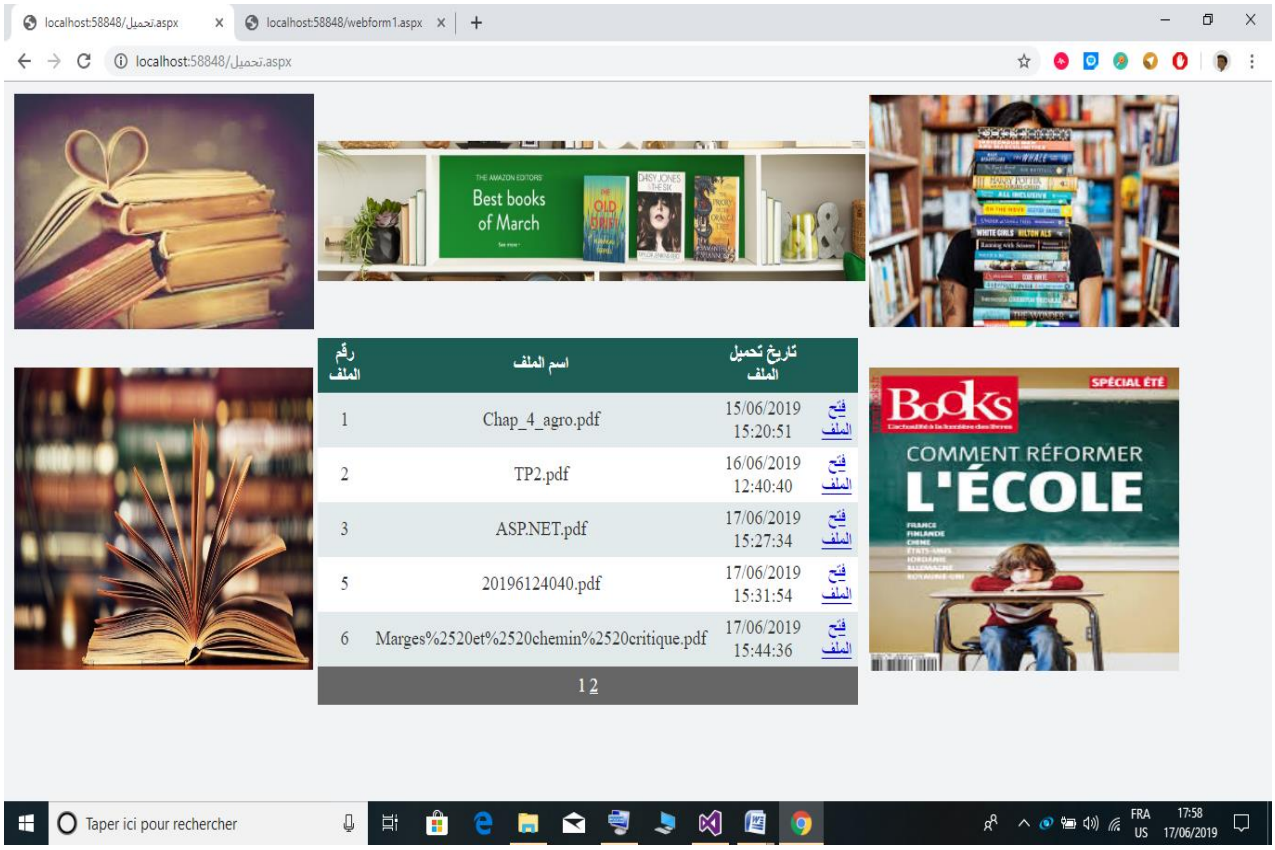

Figure 4.28 : Affichage de la collection des documents

1. Conclusion

Dans ce chapitre nous avons décrit la mise en œuvre des expérimentations menées pour valider notre travail. Nos expérimentations nous ont permis d'apprécier la performance de la recherche d'information.

Conclusion générale

Conclusion générale :

La recherche d'information (RI) est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche et la sélection d'information [Salton 1968]. Depuis sa création les chercheurs n'ont pas cessés d'affronter les défis qui s'accroissent de plus en plus que ce soit au niveau volume et variété de l'information. Ainsi, Internet grâce à ses capacités de diffusion et d'accès à l'information à ouvert de nouveaux axes de recherche pour satisfaire le besoin des utilisateurs en matière d'information. Dans ce mémoire les principaux concepts de base qui tournent autour de la recherche d'information ont été présentés, ensuite les outils de recherche d'information sont clairement cernés avec les points qu'ils ont en communs et les points qui les différencient les uns des autres.

Notre site web RI a pour objectif de sélectionner les documents satisfaisants pour un utilisateur qui s'est exprimé à travers une requête étant en besoin d'information. Cette sélection de documents est le résultat d'une série de techniques et méthodes employées.

En perspective, depuis les modèles classiques, la normalisation de la longueur du document jusqu'aujourd'hui n'a été traité par aucune approche nouvelle de la recherche d'information. Donc des développements futurs sont attendus pour pouvoir traiter la structure interne de document.

Références bibliographiques

1. **Hernandez, N.** Ontologie de domaine pour la modélisation du contexte en recherche d'information. *thèse de doctorat en informatique.* s.l. : Université Paul Sabatier, 2006.

2. **Boubekeur, F.** Contribution à la définition de modèles de recherche d'information flexibles basés sur les CP-Nets. *thèse de doctorat en informatique.* s.l. : Université Paul, 2008.

3. **Daoud, M.** Accès personnalisé à l'information : approche basée sur l'utilisation d'unprofil utilisateur sémantique dérivé d'une ontologie de domaines à travers l'historique. *thèse de doctorat en informatique.* s.l. : Université Paul Sabatier, 2009.

4. **BOURAMOUL, ABDELKRIM.** RECHERCHE D'INFORMATION. *these de doctorat.* Constantine : Université MENTOURI , 2011.

5. **Ingwersen, Peter.** *Information retrieval interaction.* London : Taylor graham Publishing, 1992.

6. **Boughanem, M et Savoy, J.** *Recherche d'information états des lieux et perspectives.* s.l. : Hermès Sience Publications, 2008.

7. **TAMINE-LECHANI, Lynda et CALABRETTO, Sylvie.** *Recherche d'information contextuelle et web.*

8. **Damak, Firas.** Etude des facteurs de pertinence dans la recherche de microblogs. *these de doctorat.* Toulouse : s.n., 2014.

9. **Buckley, C et Salton, G.** *Term weighting approches in automatic text retrieval.* 1988.

10. **Zipf, George K.** *Hman Behaviour and the principal of least effort.* USA : s.n., 1949.

11. **Maron, M. E et Kuhns, J. L.** *On relevance, probabilistic indexing and information retrieval.* 1960.

12. **Porter, M.** *An algorithm for suffix stripping* . 1980.

13. **Mayfield, J et McNamee, P.** *Single n-gram stemming. In proccedings of the 26th annual internationnal ACM SiGIR Conference on research and development in information retrieval.* New York : s.n., 2003.

14. **Manning, C, Raghavan, P et Schtze, H.** *Introduction to information retrieval.* New York : Cambridge university press, 2008.

15. **Baeza- Yates, R. A et Ribeiro-Neto, B. A.** *Modern information retrieval.* England : Pearson education Ltd, 2011.

16. **Hammache, Arezki.** Recherche d'information: un modèle de langue combinant mot simples et mots composés. *these de doctorat.* Tizi-Ouzou : s.n.

17. **Salton, G.** *The SMART Retrieval System - Experiments in Automatic Document Processing.* New Jersey : s.n., 1971.

18. **Van Rijsbergen, C.J.** *Information retrieval.* London : Butterworth, 1979.

19. **Abbassi, Meftah.** Un modèle de reformulation des requêtes pour la recherche d'information sur le Web. *these de master.*

20. **Hachemi, Hadjira et Rimouche, Nour El Houda.** Moteur de recherche sémantique. *memoire de licence.* 2013.

21. **Robertson, S.E et Sparch, Jones.** *Relevance Weighting for Search Terms.* s.l. : Journal of The American Society for Information, 1976.

22. **Robertson., S.** *The probability ranking principle in information retrieval.* 1977.

23. **Zemirli, Nesrine.** Vers le développement d'un système de recherche d'information personnalisé integrant profile d'utilisateur. *thèse de doctorat.* Université Paul Sabatier : s.n., 2004.

24. **Bruce Croft, W, Turtle, Howard R et Lewis, David D.** *The Use of Phrases and Structured Queries in Information Retrieval.* 1991.

25. **Rocchio, J.** *Relevance feedback information retrieval. .* 1971.

26. **Harman, D.** *Relevance feedback revisited.* Proceedings of ACM SIGIR : s.n., 1992.

27. [Baeza-Yates and Ribeiro-Neto,]Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 1999

28. Baziz.M " indixation conceptuelle Guidée par ontologie pour la recherche d'information "thése de doctorat decembre 2005 pages 42-43.

29. Bordogna et al., : Flexible Querying of Structured Documents. Proceedings of the Fourth International Conference on Flexible Query Answering Systems(FQAS), 2000.

30. Boucham.S, Une approche basée Ontologies pour l'indexation automatique et la recherche d'information Multilingue, mémoire de magister, Université M'hamed Bougara Boumerdes, 2009.

31. Bouramoul A " recherche d'information contextuelle et semaintique sur le web " thése pour l'optention du grade de docteur en science ,2011

32. Caro, Budi Yuwono, Savio L.Y. Lam , Jerry H. Ying, Dik L. Lee, A World Wide Web Ressource Discovery System, 1997.

33. Kohonen et al, Self-Organisation and Associative Memory3rd Edition, Springer Verlag, Berlin, pages : 80-89, 1989.

34.Elsa NEGRE CAHIER DE LAMSAD Université dauphine paris (Laboratoire d'Analyses et Modélisation de Systèmes pour l'Aide à la Décision) avril2013

35.Salton.G, Automatic text processing: The transfromation, analysis and retrieval of information by computer. Addison-Wesley publishing, pages : 85-92, 1989.

36.Xavier Tannier modèle de RI et évaluation mémoire de master 2(Indexation et Recherche d'Information) université paris-sud11

.