Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

**République Algérienne Démocratique et Populaire**
**Université Abou Bakr Belkaid– Tlemcen**
**Faculté des Sciences**
**Département d'Informatique**

**Mémoire de fin d'études**

**pour l'obtention du diplôme de Licence en Informatique**

*Thème*

# Réalisation d'un Système de Recherche d'Information

**Réalisé par :**

- **M^lle. AMIMER HANANE**
- **M^lle. CHEKROUN FADIA**

Présenté le 26 Mai 2015 devant la commission d'examination composée de MM.

- Mr. Bentaallah Mohamed (encadreur)
- Mr. Smahi Ismail (Examinateur)
- Mr.Abdellaoui Ghouti (Examinateur)

**Année universitaire: 2014-2015**

# Remerciement

*Hamdelillah ...*

*Nous remercions* ***DIEU*** *le tout puissant de nous avoir donné le courage, la patience et la force morale et physique afin de pouvoir accomplir ce travail.*

*On tient tout d'abord à remercier notre encadreur Mr.* ***Mohamed Amine Bentaallah*** *de nous avoir acceptés pour effectuer ce projet de fin d'études, pour son temps précieux, ses conseils et sa disponibilité tout au long du déroulement de ce travail.*

*Nous exprimons également nos remerciements à les membres du jury Mr.* ***Ismail Smahi*** *et Mr.* ***G.Abdellaoui****, nos remerciements les plus vifs pour avoir acceptés d'honorer par leur jugement notre travail.*

*Nous adressons nos remerciements à tous les professeurs, pour leurs conseils et leurs critiques qui ont guidé nos réflexions durant nos recherches.*

*Nous souhaitons exprimer nos profondes gratitudes à nos parents qui nous ont supportées tout au long de notre projet, ainsi que toute la famille, les amis pour leur soutien indéfectible.*

*A toutes les personnes qui nous ont aidées nous présentons nos remerciements, nos respects et nos gratitudes, et surtout notre ami Zineb, nous lui souhaitons la bonne chance dans ses études.*

# dédicace

Que ce travail témoigne de mes respects :

À mes soutiens moraux, source de joie et de bonheur, ceux qui ont toujours sacrifié pour me voir réussir, mes très chères parents que j'adore .Aucune dédicace ne pourrait exprimer mon respect, ma considération et mes profonds sentiments envers eux. Je prie le Dieu les procure bonne santé et longue vie.

A mes sœurs :

Naval et son mari Farid, Warda et son mari Karim.

A mon frère :

Zouheir et sa femme Saliha.

Ainsi que mes neveux et mes nièces :

Ines, Sarah, Rayhana, Rayane, Hadil.

A mon binôme Fadai, qui me comprend et me encourager de réaliser ce travail ensemble, qui partage avec moi les joies et les difficultés relatives au suivi de ce projet.

A tous mes amis et notamment Fatima, Fatna, Aicha, Zahra, Ghizlen, Houaria. Je ne peux trouver les mots justes et sincères pour vous exprimer mon affection et mes pensées, vous êtes pour moi des sœurs.

A tous ceux qui de loin ou de prés m'ont soutenu, d'une manière ou d'une autre pour que ce projet soit possible, trouvent ici l'expression de ma profonde reconnaissance.

*Hanane*

# dédicace

Je dédie ce modeste travail à :

A mes parents .Aucun hommage ne pourrait être à la hauteur de l'amour Dont ils ne cessent de me combler. Que Dieu leur procure bonne santé et longue vie.

A mon fiancé que je remercie tendrement pour son soutien moral, son aide, ses conseils et ses encouragements qui m'ont facilité ce travail ;

A mes sœurs Wafaa et Fatima Kawter qui me sont chères ;

A mon petit cher frère Mohammed Réda.

A ma tante et ma deuxième mère « Kouria», pour leur soutiens ;

A mon grand-père que j'aime beaucoup, pour toutes ses prières intarissables.

A mes chères cousines : Imène, Djahida, Hanane, Marwa, et Mayada.

A mes oncles, mes tantes, mes cousines, mes cousins et toute ma famille ;

A mon beau père « Mr. Abdelkrim », et toute ma belle famille ;

A ma binôme et meilleure amie "Hanane " qui ma supporté tout au long de notre projet ;

A mes meilleurs amies Zahra, Ghizlane et Kouaria et Ikram ;A tous ceux que je porte dans mon cœur, et à tous ceux qui ont contribué de près ou de loin pour que ce projet soit réalisé, je vous dis merci.

FADIA

# Table des matières

# Table des matières

# Introduction générale

La recherche documentaire a été initialement développée pour gérer la littérature scientifique énorme qui s'est développée. De nombreux universitaires, des entreprises, et les bibliothèques publiques ont besoin d'un mécanisme pour faciliter l'accès à des livres, des revues et d'autres documents.

Avec le développement des technologies, aujourd'hui l'information joue le rôle primordial dans le quotidien des individus et dans l'essor des entreprises, le volume d'information dans tous les domaines ainsi que le nombre d'utilisateurs sont toujours en croissance. Par conséquent, il est de plus en plus difficile de localiser précisément ce que l'on recherche dans cette masse d'information, le domaine par excellence qui s'intéresse à répondre à ce type d'attente c'est la recherche d'information (RI).

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 défit est de pouvoir, parmi le volume important de documents disponibles, trouver ceux qui correspondent au mieux à l'attente de l'utilisateur, l'opérationnalisation de la RI est réalisée par des outils informatiques appelés Systèmes de Recherche d'Information (SRI) **[01].**

SRI a 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). Cette comparaison est réalisée par des modèles de recherche, la majorité de ces modèles représentent les documents et les requêtes par une liste de mots clés pondérés. Le calcul de la pertinence est obtenu à partir des poids censés refléter l'importance des termes dans les documents.

Notre travail consiste à réaliser un SRI qui répond à un besoin d'utilisateur par une étude de différentes notions de contexte en recherche d'information. En utilisant des corpus, des jugements de pertinence et des critères d'évaluation adaptés pour la recherche d'information.

Notre mémoire est structurée en deux chapitre, le premier chapitre est un chapitre généraliste sur la RI, dans lequel on va présenter les principaux concepts de la RI, les modèles de RI, ainsi que les approches de l'évaluation des SRIs. Nous commençons tout d'abord par donner quelques définition, puis nous décrivons les étapes d'un processus de RI, nous passerons ensuite à les différent modèles de recherche et le domaine d'application de la RI, enfin nous présenterons les plus importantes mesures utilisées pour évaluer un SRI. Le second chapitre se porte sur la réalisation et la conception de notre système, on donne une description de l'environnement de travail et les déférentes étapes qu'on a suivies pour réaliser notre SRI, afin de répondre a un besoin d'information et retrouver tous les documents pertinents a une requête utilisateur.

# Chapitre I

# Recherche d'Information

## 1. Introduction :

La recherche d'information (RI), est une branche en informatique apparue comme une discipline de recherche, afin de résoudre le problème lié de l'accès à l'information contenue dans une grande masse de documents au moyen d'un système de recherche d'information (SRI) qui consiste à construire une représentation des documents – requête, et établir une comparaison entre ces deux représentations pour avoir des documents pertinents. Cette comparaison est basée sur des modèles qui fournissent une base d'évaluation afin de mesurer les réponses du système avec les réponses idéales que l'utilisateur espère recevoir.

Le but de ce chapitre est de présenter la RI et les concepts de base de la RI. Nous nous intéressons au processus de RI, puis nous présentons le SRI et les mesures de son évaluation.

## 2. Définition de la recherche d'information :

Salton a définit la recherche d'information comme « *un domaine qui traite de la représentation, du stockage, de l'organisation et de l'accès à l'information* ». **[02]**

D'après Van Rijsbergen « *la RI consiste à restituer les documents qui peuvent être pertinents par rapport au besoin d'information exprimé dans la requête* ». [**03]**

Une autre définition que « *la RI est l'ensemble des méthodes, procédures et techniques ayant pour objet d'extraire d'un document ou d'un ensemble de documents les informations pertinentes* ». [**04]**

D'après ces définitions, on conclut que la RI est un domaine qui s'intéresse à répondre à un besoin d'information, à l'aide d'un ensemble des méthodes qui facilitent la recherche dans une grande base de documents.

## 3. Bref Historique de la RI: [05]

Le domaine de la RI remonte au début des années 1940, peu après l'invention des ordinateurs, nous allons retracer brièvement son évolution a travers le temps :

-1940: Apparition des SRI, focalisation de la RI sur les applications dans des bibliothèques.

-1950: Apparition du modèle booléen et l'élaboration de petites expérimentations sur des petites collections de documents.

-1960 et 1970: Apparition du système SMART, Développement d'une méthodologie d'évaluation de système et conception de corpus de test(CACM).

-1980: Développement de l'intelligence artificielle, ainsi on tentait d'intégrer des techniques de l'IA en RI (système expert).

-1990 et 1995: L'apparition d'internet, la RI a été modifié et sa problématique plus élargie.

## 4. Notion de base de RI :

- 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**. [06]**

- Corpus :

Est un ensemble d'informations exploitables et accessibles par un utilisateur ou c'est l'ensemble de documents dans lequel l'utilisateur cherche une information.

- Besoin d'information :

Un besoin en information est une représentation mentale de ce que l'utilisateur souhaite rechercher. Ce besoin est représenté sous forme d'une requête**. [07]**

- Requête :

C'est une expression du besoin de l'utilisateur, elle représente l'interface entre le SRI et l'utilisateur.

- Pertinence :

La notion de pertinence est le critère le plus important dans la RI et l'objet de tout système de recherche d'information, cette notion n'est pas facile a définir car elle fait intervenir plusieurs notions, mais la définition la plus simple : « *La pertinence est la correspondance entre un document et une requête, ou encore un degré de relation du document à la Requête et une mesure d'utilité du document pour l'utilisateur* » **[08].**
La pertinence est liée au deux niveaux :

Niveau système : le système mesure un degré de pertinence, une valeur de similitude entre un document et une requête.

Niveau utilisateur : pour l'utilisateur la pertinence correspond à la satisfaction d'ensemble des documents restitués par le SRI. Deux utilisateurs peuvent juger différemment un même document renvoyé pour une même requête.

## 5. Processus de RI :

Le processus de RI consiste à mettre en correspondance et à calculer le degré d'appariement des représentations interne des documents et de la requête. Les documents qui retournées a l'utilisateur (documents dits pertinents), sont ordonnée dans une listes par ordre décroissant de degré de pertinence .Afin d'améliorer les résultats de la recherche, le système peut être doté un mécanisme d'amélioration de la requête par la reformulation.

Donc le Système de Recherche d'Information (SRI) crée une interface entre la base documentaire, ou collection de documents, et les utilisateurs qui recherchent des informations contenues dans cette base, ce processus est composé de trois étapes essentielles :

- L'indexation.
- L'appariement document-requête.
- Reformulation de requête.

Le système de recherche d'information intègre ces trois fonctions principales par un processus en U. Comme illustré dans **la figure 1.1** ,Ce processus manipulé d'un coté des collections des documents et d'un autre coté le besoins d'information exprimés par l'utilisateur sous forme d'une requête , après le traitement des documents et requêtes (indexation) le système fait la correspondance entre les deux représentation (appariement document-requête) ,puis retourne une liste de documents pertinents sélectionnés (ce qu'on appelle pertinence système), comme il y'a aussi la pertinence utilisateur ,tel que l'utilisateur qui peut indiquer au système les documents qu'il juge réellement importants.

A l'aide de ces jugements, le système doit être capable de construire automatiquement une nouvelle requête (reformulation de requête).

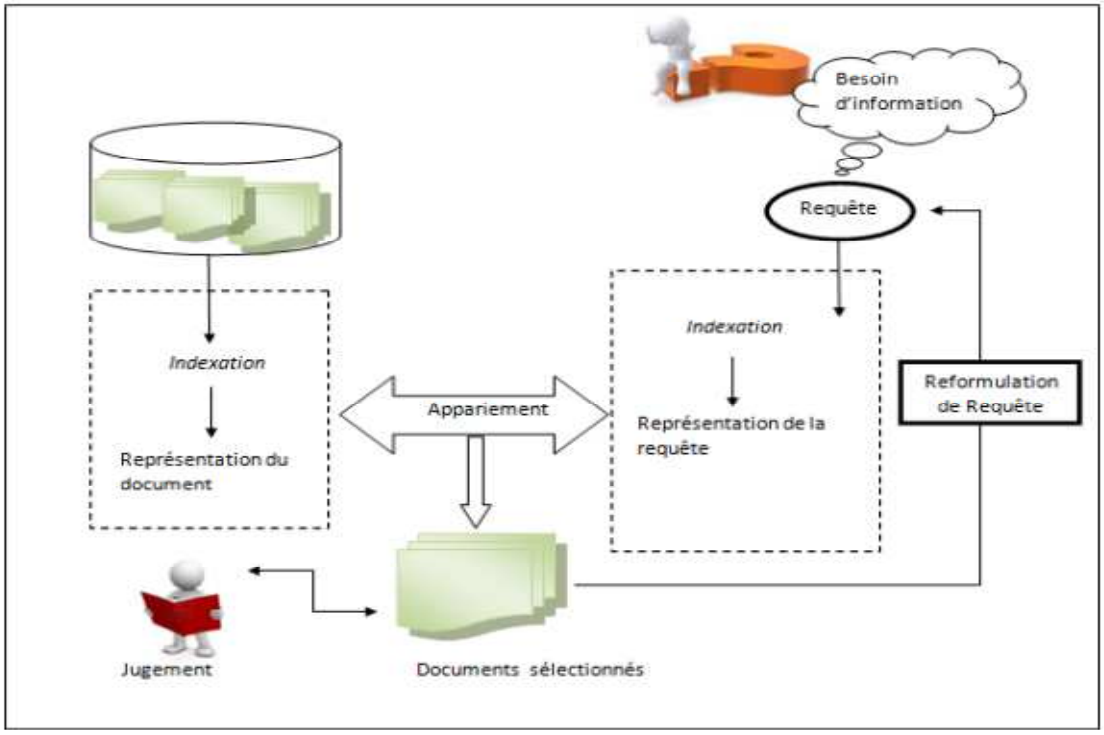



**Figure 1.1 Processus en U de la RI**

## 5.1 Indexation:

Une étape très importante doit s'effectuer avant l'étape de recherche effective de l'information. Cette étape consiste à analyser les documents et la requête d'utilisateur, afin de construire des index qui faciliteront le processus de recherche, Cette opération est appelée indexation .Les performances et la qualité de réponse de système dépendent de cette phase.

Vue de son importance, et soucieux de bien la réaliser, les développeurs des SRI ont proposé plusieurs manières de procéder. Les principales manières sont :

✓ **Manuelle :**

Dans ce cas, chaque document est analysé par un spécialiste du domaine ou par un documentaliste. Après la lecture des documents, ce spécialiste détermine, selon des connaissances, les mots-clés qui lui semblent les plus adéquats pour représenter le contenu du document. Ce mode d'indexation est fondé sur le jugement humain, Il se caractérise par sa profondeur, sa cohérence et sa qualité. **[08]**

L'indexation manuelle à l'avantage d'assurer une meilleure correspondance entre les documents et les termes choisis par les indexeurs pour les représenter, mais l'inconvénient majeur de cette méthode d'indexation est que des termes différents peuvent être sélectionnés par des indexeurs différents. Il peut même arriver qu'une personne, à des moments différents, indexe différemment le même document**. [06]**

✓ **Automatique :**

Ce type d'indexation est actuellement le plus répandu car il détecte automatiquement les termes les plus représentatifs du contenu du document, il comprend un ensemble de traitements sur les documents.

On y distingue : l'analyse de texte du document mot à mot, extraire les mots vides qui ne jouent qu'un rôle syntaxique, éliminer les mots qui apparaissent trop souvent et qui n'ont aucun intérêt, pondérer les termes et finalement la création de l'index.

✓ **semi-automatique :**

Appelée aussi l'indexation supervisée. c'est un rapport qui combine les deux types d'indexation manuelle et automatique, ainsi l'indexation automatique fait les premiers éléments de l'indexation, puis le spécialiste du domaine corrige manuellement les informations sélectionnées par l'indexation automatique.

Le résultat de l'indexation est un ensemble de termes définissant ce qui est appelé le ***langage d'indexation***. On peut distinguer deux types de langage d'indexation :

- langage contrôlé :

Un langage d'indexation contrôlé est construit à partir d'un professionnel qui doit choisir un ou plusieurs descripteurs pour représenter le document.

- langage libre :

Appelé aussi langage naturel, un langage d'indexation libre est construit à partir des termes extraits du document analysé.

L'indexation doit traduire le contenu conceptuel d'un document. Une bonne indexation devrait donc recenser les thèmes abordés de manière précise et exhaustive après avoir passé par une suite d'étapes comme le montre **la figure 1.2** :

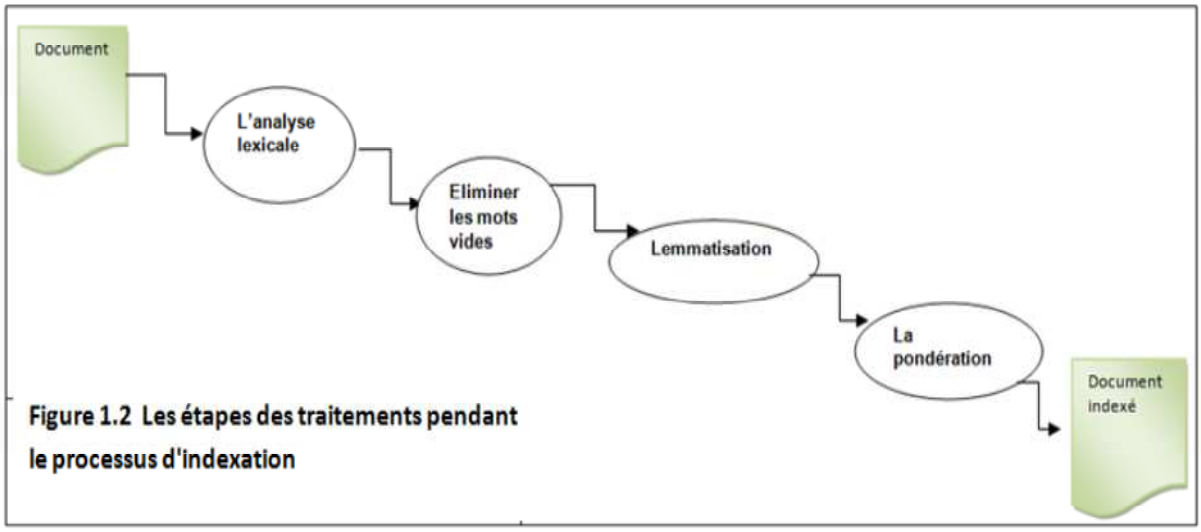



**Figure 1.2 Les étapes des traitements pendant le processus d'indexation**

- **Analyse lexicale (segmentation) :**

C'est un processus qui convertit le texte d'un document en un ensemble de termes ou un terme est un radical ou une unité lexicale. Cette analyse permet de reconnaître les espaces de séparation, des chiffres, des mots et des ponctuations. [**09]**

- **Elimination des mots vides :**

L'élimination des mots vides est l'une des étapes du processus d'indexation permettant d'améliorer la fiabilité d'un SRI au sens de qualité logiciel (temps d'exécution) et de performance **[10].**

Elle consiste a évité les mots vides (pronom personnel, prépositions, articles, mots mathématiques (appartenir, contenir, inclure.)….) et choisir seulement les termes significatifs qui représentent au mieux un document donné. Afin d'éliminer ces mots de

force, on utilise une liste, appelée stop-List (ou parfois anti-dictionnaire) qui contient tous les mots qu'on ne veut pas garder.

L'élimination de ces mots permet de réduire l'index, on gagne alors en espace mémoire, mais aussi le non traitement des mots vides permet de réduire le temps d'exécution.

- **Radicalisation :**

L'idée qui conduit à utiliser la lemmatisation est de pouvoir indexer un ensemble de mots par un seul mot qui représente le même concept.

En effet, on remarque que beaucoup de mots ont des formes différentes, mais leur sens reste le même ou très similaire et notamment dans le cas des mots conjugués. Ces mots ont la même racine (lemme). Ainsi, on arrive à éliminer les terminaisons des mots, et garder seulement la racine, on a donc une forme identique pour eux. Plusieurs méthodes sont utilisées pour retrouver la racine lexicale d'un mot tel que l'« algorithme de Porter », la troncature et la méthode des n-grammes.**[09]**

Par exemple, selon un algorithme adapté de Porter en 1980 **[11].** Pour la langue anglaise, le mot "*information*" est transformé en racine "*inform*".

Le tableau qui suit illustre un exemple de lemmatisation en utilisant différentes techniques de lemmatisation.

| ***Unité descriptive*** | ***Représentation*** |
|---|---|
| Tri-grammes | A **co ;mpu ;ter ;** is ; a ;gen ;era ;l p ;urp ;ose ; de ;vic ;e t ;hat ; ca ;n b ;e p ;rog ;ram ;med ; to ; ca ;rry ; ou ;t a ; se ;t of ; ar ;ith ;met ;ic ;or ;log ;ica ;l o ;per ;ati ;ons ; au ;tom ;ati ;cal ;ly ;. |
| Mots | A ; **computer ;** is ; a ; general ; purpose ; device ; that ; can ; be ; programmed ; to ; carry out ; a ; set ; of ; arithmetic ; or ; logical ; operations ; automatically. |
| Lemme | A ; **computer ;** is ; a ; gener ; purpos; devic ; that ; can be ; program ; to ; carr ; out ; a ; set ; of ; arithmet ; or ; logic ; oper ; automat. |

**Tableau 1.1 : Les différentes représentations de contenu du texte dans l'indexation.**

- **Pondération :**

La pondération est l'une des fonctions fondamentales en RI. Elle est la clé de voûte de la majorité des modèles et approches de RI proposés depuis les années 1960. Le poids représente l'importance d'un terme dans un document **[06]**. On distingue deus types de pondération :

➢ **Pondération locale :**

Indique l'importance locale du terme dans un document. La fonction de pondération locale la plus utilisée est la suivante :

- $tf_{ij}$ (terme frequency) : Correspond au nombre d'occurrences de chaque terme $t_i$ dans chaque document $D_j$.

➢ **Pondération globale :**

**-idf (**Inverse o*f* document frequency) **:**

Elle indique une représentation globale du terme dans l'ensemble des documents de la collection, un poids plus important est donné aux termes qui apparaissent dans peu de documents. Un facteur de pondération globale **idf** dans l'ensemble de documents a été introduit. Il peut être calculé selon :

$$Idf(w) = \log (N/ (df(w)+1)) +1$$

Où **df** (w): est le nombre de documents contenant le terme w.

**N :** le nombre total de documents dans la collection.

De manière générale, la majorité des méthodes de pondération sont construite par la combinaison de deux facteurs. Un facteur de pondération locale, quantifiant la représentativité locale d'un terme dans le document, et un second facteur de pondération globale, mesurant la représentativité globale du terme vis-à-vis de la collection des documents. La méthode de pondération la plus utilisé est la méthode TFIDF.

**- TF*IDF** :

Est une mesure de bonne approximation de l'importance d'un terme dans un document, particulièrement dans des corpus de documents de tailles homogènes. Cette mesure a eu en revanche un succès très limité dans les corpus de tailles très variables.

$$\text{Tf*Idf (w)} = \text{tf (w)} * (\log (N/ (\text{df (w)} +1)) +1)$$

Des nouvelles méthodes ont été effectuées pour améliorer cette fonction de pondération comme la méthode TFC.

**-TFC :**

La pondération ***TF*IDF*** ne prend pas en considération le fait que les documents peuvent être de différentes longueurs, ainsi la pondération ***TFC*** est semblable au ***TF*IDF*** à la différence que la ***TFC*** emploie la longueur normalisée**.[12]**

$$TFC_{ij} = \frac{f_{ij} * df_i}{\sqrt{\sum_{k=1}^{M} (f_{kj} * df_k)^2}}$$

**-LTC :**

La pondération ***LTC*** est une approche légèrement différente, qui emploie le logarithme de la fréquence d'une unité lexicale, de ce fait elle réduit les effets de grandes différences dans les fréquences**. [12]**

$$Ltc = \frac{\log(f_{ij} +1) * df_i}{\sqrt{\sum_{k=1}^{M} (\log(f_{kj}+1) * df_k)^2}}$$

## 5.2 L'appariement document-requête :

Grâce à la relation d'appariement, il est possible de rechercher parmi les différents documents transformés, ceux qui répondent le mieux à une requête utilisateur. Le

système retourne à l'utilisateur les documents qui correspondent à sa requête après avoir calculer un score de correspondance entre la représentation de chaque document et celle de la requête. Ce score est calculé au moyen d'une fonction de similarité appelée RSV(D,Q) (Retrieval Statu Value) , où D est un document et Q la requête .

## 5.3 La reformulation de requête :

La reformulation de requêtes est un processus ayant pour objectif de générer une nouvelle requête plus adéquate que celle initialement formulée par l'utilisateur en rajoutant de nouveaux termes et/ou supprimant des termes inutiles. Cette reformulation permet de coordonner le langage de recherche (utilisé par l'utilisateur dans sa requête), et le langage d'indexation des documents.

On distingue principalement deux approches de reformulation de requête : une approche basée sur un processus automatique et une autre, basée sur un processus interactif. Nous allons détailler dans les paragraphes suivants ces deux approches et nous présentons les principaux travaux développés. **[13]**

- **Expansion automatique des requêtes :**

L'expansion directe de la requête consiste à rajouter à la requête initiale des termes issus de ressources linguistiques existantes ou bien de ressources construites à partir des collections. Plus précisément, au niveau des ressources linguistiques, le but est d'utiliser un vocabulaire contrôlé issu de ressources externes. On peut alors utiliser des ontologies linguistiques (citons par exemple Word net). [**14]**

- **La reformulation interactive :**

Dans une reformulation interactive, l'utilisateur joue un rôle actif. A l'inverse de la reformulation automatique, ce sont le système et l'utilisateur qui sont, ensemble, responsables de la détermination et du choix des termes candidats à la reformulation. Le système joue un grand rôle dans la suggestion des termes, le calcul des poids des termes et l'affichage à l'écran de la liste ordonnée des termes. L'utilisateur examine cette liste et décide le choix des termes à ajouter dans la requête. C'est donc l'utilisateur qui prend la décision ultime dans la sélection des termes. **[15]**

## 6. Système de recherche d'information (SRI) :

### 6.1. Définition de SRI :

Toutes les définitions des systèmes de recherche d'information expriment le même principe :

Selon Tebri **[03],** un SRI est un ensemble de programmes informatiques qui a pour but de sélectionner des informations pertinentes répondant à des besoins utilisateurs, exprimés sous forme de requêtes.

Alors que Chein **[16]** décrit un SRI comme étant un système informatique dont le but est d'aider un utilisateur à trouver des documents contenant des informations pertinentes pour un besoin d'information exprimé par une requête au système.

Cependant, nous gardons celle donnée par Mathias **[17],** qui est la plus simple: Un Système de Recherche d'Information est un système informatique qui facilite l'accès à un ensemble de documents, pour permettre de retrouver ceux dont le contenu correspond le mieux à un besoin d'information d'un utilisateur.

Tout Système de Recherche d'Information a deux objectifs principaux :

-Le premier est de retrouver tous les documents pertinents pour une requête utilisateur.

-Le deuxième est de rejeter les documents jugés non pertinents.

### 6.2. Modélisation de système de recherche d'information :

Un SRI peut se baser sur plusieurs grands types de modèles afin de mesurer l'adéquation entre une requête et un ensemble de mots-clés issus de l'indexation d'un document.

Un modèle de RI a pour rôle de fournir une formalisation du processus de RI, il doit accomplir plusieurs rôles dont le plus important est de fournir un cadre théorique pour la modélisation de cette mesure de pertinence. [**06]**

Les modèles de RI peuvent être classés en trois principales classes qui sont :

- **Le modèle Booléen ou ensembliste :**

Le modèle booléen repose sur la manipulation des mots clés. D'une part, un document est représenté par une conjonction de mots clés, d'autre par une requête (R) est représentée par une expression logique composée de mots connectés par des opérateurs booléens (ET, OU, NON). Par exemple : $d = t1 \wedge t2 \wedge t3 ... \wedge tn$ .

La formulation de la requête se base sur les trois opérateurs booléens :

• La conjonction ($\wedge$) exige que les termes soient présents simultanément dans la description d'un document.

• La disjonction ($\vee$) exige qu'au moins un des termes soit présent dans la description des documents à retourner.

• La négation non ($\neg$) utilisée pour écarter les documents qui contiennent un terme. **[04]**

Par exemple : $q = (t1 \vee t2) \wedge \neg (t3 \vee t4)$

Le modèle booléen utilise le mode d'appariement exact {0 ,1} c.à.d.

$$\begin{cases} RSV(d, q) = 1 \quad \text{si} \quad q \in d \\ RSV(d, q) = 0 \quad \text{sinon} \end{cases}$$

Donc il ne restitue que les documents répondant exactement à la requête ce qu'on appelle l'ensemble idéal. Ce modèle est très largement utilisé, aussi bien pour les bases de données bibliographiques grâce a la simplicité de la mise en œuvre, mais son inconvénient, est la complexité de la formulation de requêtes ainsi que l'impossibilité d'ordonner les documents retournés.

- **Modèles vectoriels :**

Le modèle vectoriel (nommé aussi VSM pour Vector Space Model), a été popularisé en 1971 par Salton **[18]**. Ce modèle propose de représenter les documents et les requêtes par des vecteurs d'indexation dans un espace engendré par les termes d'indexation. Le modèle vectoriel représente les requêtes et les documents sous forme de vecteurs dans un même espace vectoriel. La mesure de similarité entre le document représenté par un vecteur $d = (d1, d2 \ldots dn)$, où di est le poids d'un terme i dans le document, et la requête définie par $q = (q1, q2 \ldots qn)$ où qi est le poids (souvent 0 ou 1 selon que le terme appartient ou pas à la requête) du terme i dans la requête, est estimée

selon les propriétés et les mesures populaires issues de la théorie des espaces vectoriels.**[10]**

Un des principaux inconvénients de ce modèle réside dans l'obligation pour un texte de contenir au moins un des mots de la requête pour pouvoir être retrouvé (car les dimensions de l'espace vectoriel étant orthogonales, les termes sont considérés comme indépendants) mais il reste le model le plus simple à utiliser et facile à implémenter avec l'algèbre linéaire.

Une fonction de correspondance est calculée pour mesurer le degré de similarité entre le vecteur document et le vecteur requête, les fonctions de similarité les plus utilisées sont :

| *Mesures* | *Formules* |
|---|---|
| **Le produit scalaire** | $RSV(q, di)=\sum_{j=1}^{\lvert T\rvert} w_{qj} \cdot w_{ij}$ |
| **La mesure de cosinus** | $RSV(q, di)=\frac{\sum_{j=1}^{\lvert T\rvert} w_{qj} \cdot w_{ij}}{\sqrt{\sum_{j=1}^{\lvert T\rvert} w_{qj}^{2} \sum_{j=1}^{\lvert T\rvert} w_{ij}^{2}}}$ |
| **La mesure de Dice** | $RSV(q, di)=\frac{2 * \sum_{j=1}^{\lvert T\rvert} w_{qj} \cdot w_{ij}}{\sqrt{\sum_{j=1}^{\lvert T\rvert} w_{qj}^{2}+\sum_{j=1}^{\lvert T\rvert} w_{ij}^{2}}}$ |
| **La mesure de Jaccard** | $RSV(q,d_i)=\frac{\sum_{j=1}^{\lvert T\rvert} w_{qj} \cdot w_{ij}}{\sum_{j=1}^{\lvert T\rvert} w_{qj}^{2}+\sum_{j=1}^{\lvert T\rvert} w_{ij}^{2}-\sum_{j=1}^{\lvert T\rvert} w_{qj} \cdot w_{ij}}$ |

**Tableau 1.2 : les mesures de similarité utilisée dans le modèle vectoriel**

La **figure 1.3** illustre une représentation vectorielle dans un espace composé de deux termes, avec deux documents (d1, d2) et une requête (q), plus l'angle formé par le document est la requête est petit, plus le cosinus de cette angle est grand et par la suite

ce document est considéré comme étant le plus pertinent à la requête, comme le document (d1) avec la requête (q).

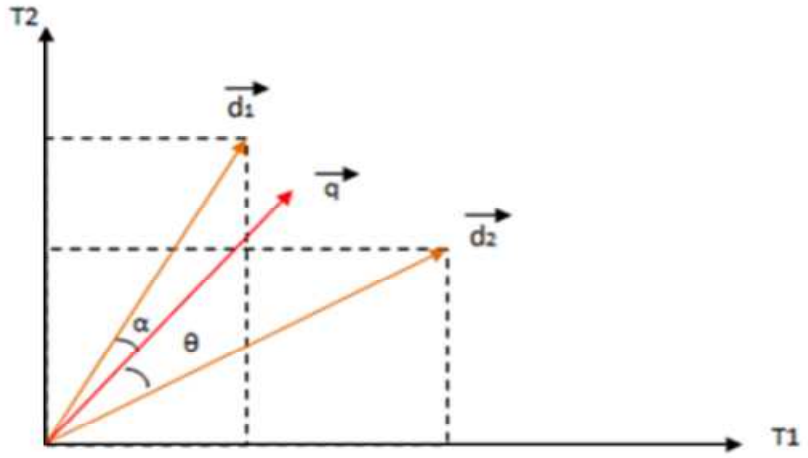



**Figure 1.3 Représentation vectorielle de deux documents (d1, d2) et une requête (q)**

- **Modèle probabiliste :**

Le modèle probabiliste aborde le problème de la recherche d'information dans un cadre probabiliste. Ce dernier propose de modéliser le processus de sélection des documents dans un SRI en se basant sur la théorie des probabilités. Le principe de base du modèle probabiliste consiste à présenter les résultats de recherche d'un SRI dans un ordre basé sur la probabilité de pertinence d'un document vis-à-vis d'une requête **[08].** Le modèle probabiliste tente d'estimer la probabilité que le document appartient à la classe des documents pertinents (ou non pertinents) en utilisant deux probabilités conditionnelles : Un document est alors sélectionné si la probabilité qu'il soit pertinent à Q, notée *P(R/d)*, est supérieure à la probabilité qu'il soit non pertinent à Q, notée *P (NR/d)*. Le score d'appariement entre le document *d* et la requête Q, noté *RSV (d, Q)* est donnée par :

$$RSV(d, Q) = \frac{p(R/D)}{p(NR/D)} \quad [20]$$

## 7. Domaine d'application de la RI :

La recherche d'information est un domaine très large, et qui peut être utilisée dans plusieurs types d'application afin de répondre à un besoin en information. Comme le montrer **le tableau 1.3 : [21]**

| *Catégorie* | *Description* | *Exemple de la requête* |
|---|---|---|
| Ad hoc Recherche | Retrouver les documents pertinents dans une collection fixe | Find documents which tell me about investment strategies. |
| Question /Réponse | Extraire des réponses dans les documents récupérés | Who is the prime minister of Australia ? |
| Annuaire | Navigation dans une Web page spécifique | Where is the ELSNET home page ? |
| Diffusion sélective d'information | Contrôlez un flot de documents correspondant à un profil | Send me any new information on high tech companies. |
| Classification de documents | Regroupements automatique de documents | Find the natural groupings in this set of scientific publications. |
| Catégorisation de documents | Affecter un document à une categorie prédéfinie | Classify incoming books according to their Dewey decimal category. |
| Synthèse de documents | Extraire l'information à partir des documents retrouvés | Construct a personalized travel guide for my visit to Athens in July 2000. |
| Recherche dans la base de données | Extraire des enregistrements à partir d'une base de données structurée | Find books where author=Salton and year=2001 |

**Tableau 1.3 : Exemples de différents types d'application de RI**

## 8. Mesures d'évaluation d'un SRI :

Un SRI capable de trouver tous les documents pertinents et rejeter tous les documents non pertinents pour une requête utilisateur est un système performant.

L'évaluation des systèmes peut être abordée selon deux angles : l'efficience qui est liée au rendement (rapidité et/ou quantité de ressources utilisées) et l'efficacité qui est lié à la qualité du résultat.

**Efficience :**

L'efficience mesure la qualité de la recherche en termes de critères liés au déroulement pratique d'une session de recherche. Parmi ces critères, on cite notamment: le délai de réponse, le nombre d'entrée-sortie sur disque, la taille de l'index**. [21]**

**Efficacité :**

L'efficacité mesure la pertinence de la recherche. Les performances liées à l'efficacité des SRI sont mesurées en comparant les documents retrouvés par le système avec les documents que l'utilisateur souhaite retrouver. A cet effet, on utilise une collection de test. **[21]**

Les mesures les plus utilisés pour évaluer les SRI sont le rappel et la précision .Comme le montre la **figure1.4:**

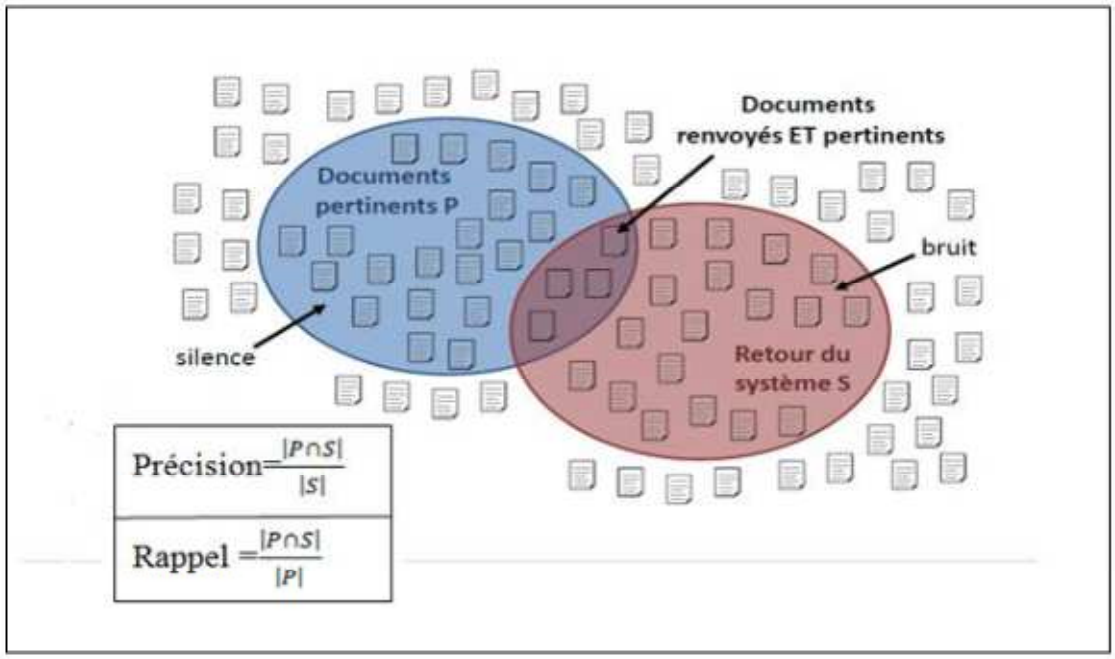



**figure1.4: Rappel / Précision**

**Le rappel :**

Le rappel mesure la proportion des documents pertinents restitués par le système relativement à l'ensemble des documents pertinents contenus dans la base documentaire. Elle mesure la capacité du système à retrouver tous les documents pertinents répondant à une requête. **[08]**

**La précision :**

La précision mesure la proportion des documents pertinents relativement à l'ensemble des documents restitués par le système. Elle mesure la capacité du système à rejeter tous les documents non pertinents à une requête donnée par le rapport entre

l'ensemble des documents sélectionnés pertinents et l'ensemble des documents sélectionnés. [**08]**

$$\textbf{Rappel} = \frac{\textit{le nombre de documents pertinents retrouvés par le système}}{\textit{le nombre de documents pertinents dans la collection}}$$

$$\textbf{Précision} = \frac{\textit{le nombre de documents pertinents retrouvés par le système}}{\textit{le nombre de documents etrouvée par le système}}$$

Pour obtenir la courbe de rappel/précision, il faut calculer plusieurs paires de rappel et précision et les interpoler, comme le montre **la figure 1.5 [08]** :

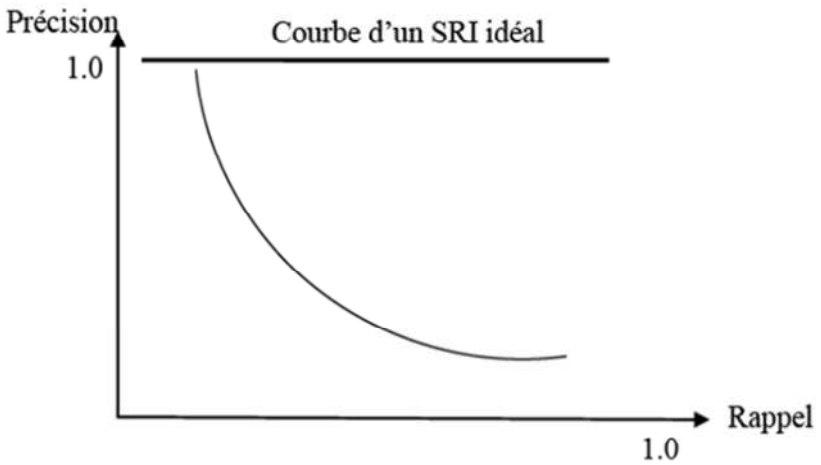



**Figure 1.5 Courbe rappel/précision**

D'autres mesures qui pourraient être d'intérêt sont généralement des mesures d'évaluation complémentaires au rappel et à la précision:

**Bruit / Silence :**

Ce sont deux notions sont complémentaire à la précision et le rappel, les valeurs de ces mesures sont comprise entre 0 et 1. Ils sont définis par :

$$B = 1 - P \quad et \quad S = 1 - R$$

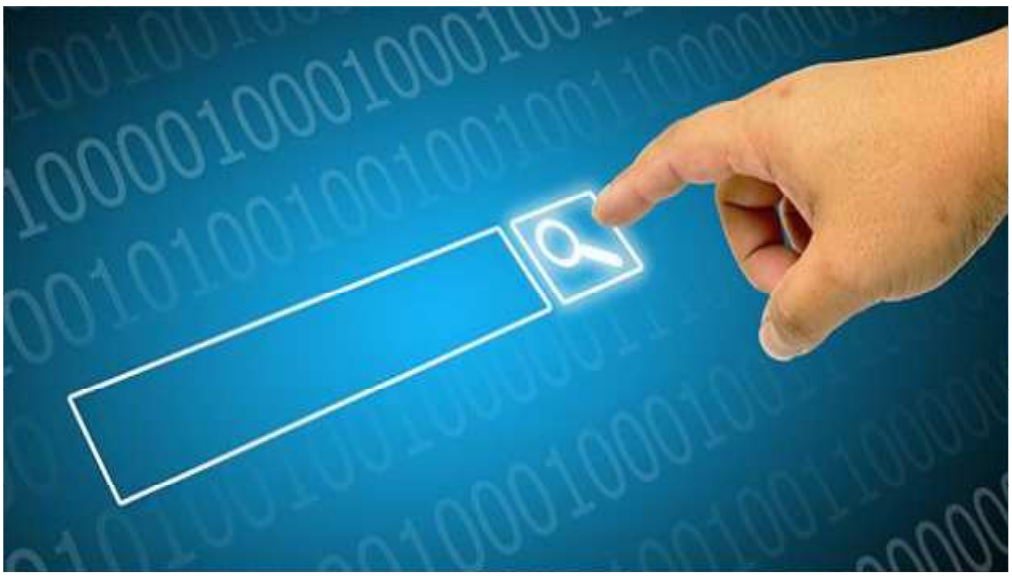
Où P est la précision du SRI et R est le rappel du SRI. **[10]**

## 9. Conclusion :

Nous avons illustré dans ce chapitre les principales notions de la recherche d'information (RI), nous avons aussi expliqué les étapes d'un processus en U de recherche d'information, ainsi les différents modèles utilisés dans la réalisation d'un SRI, et les mesures de l'évaluation de ce dernier.


# Chapitre II

# Réalisation de l'Application



## 1. Introduction :

Un SRI manipule un corpus de documents qu'il transpose à l'aide d'une fonction d'indexation en un corpus indexé. Ce corpus lui permet de résoudre des requêtes traduites à partir de besoins utilisateur.

Un tel système repose sur la définition d'un modèle de RI qui effectue ces deux transpositions et qui fait correspondre les documents aux requêtes par une relation de pertinence.

Après avoir donné un aperçu générale de la RI, nous passons à l'implémentation de l'application. Dans ce chapitre on va se baser sur le modèle le plus utilisée qui est le modèle vectoriel pour la réalisation de notre SRI. Plus précisément, nous allons détailler les étapes suivies pour la réalisation de notre SRI en montrant les différentes techniques utilisées pour la pondération ainsi que les mesures de similarité utilisés.

## 2. Les étapes de la réalisation du système :

### 2.1 Le prétraitement :

Le prétraitement constitue la première étape dans la réalisation où on va indexer et analyser chaque document dans le corpus :

#### a. L'ouverture et la lecture de document texte :

On va ouvrir chaque fichier sous forme d'une chaine de caractère pour pouvoir appliquer les différents traitements, voici un exemple de la lecture d'un fichier :

```
INFORMATION RETRIEVAL: @
Is the process of recovering specific information from stored data .
{-Relevance Is : an important concept in information  retrieval (IR) ; }
```

#### b. Formater les mots en minuscule :

Pour que le système soit capable de connaitre tous les mots, il est nécessaire d'éliminer les majuscules, car un mot écrit en majuscule est considéré comme étant différent lorsqu'il est écrit en minuscule.

information retrieval : @
is the process of recovering specific information from stored data .
{-relevance is : an important concept in information retrieval (ir) ; }

### c. Segmentation des documents :

Dans cette étape, il s'agit d'extraire les mots contenus dans chaque fichier. Dans notre système un mot est considéré comme une suite de caractères délimités entre deux séparateurs. L'ensemble des séparateurs utilisé est montré dans la **figure 2.1** :

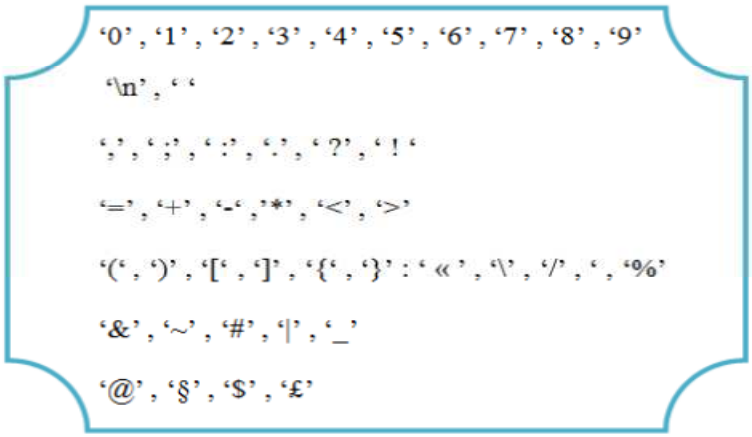



**Figure 2.1 : les séparateurs**

L'algorithme suivant illustre les étapes à suivre pour la segmentation de corpus.

```
Entrée : - corpus.
         -liste de séparateurs (délimiter).
Sortie : tableau de tokens.
Variable : chaine de caractère : ligne ;  tableau : tab
Début :
Ouvrir le corpus
Pour chaque document de corpus faire
    Line ← lire ligne
    Tant que line lu est différent EOF faire
        Tokenizer (line, délimiter)
            Tant que  il y'a des tokens  faire
            tokens ←  tokens suivant
            insérer le tokens dans  tab
            Fin tantque
            line ← lire ligne
    Fin tantque
  document suivant
 Finpour
Fin .
```

Après l'élimination des séparateurs, les mots (tokens) sont représentés comme le montre **la figure 2.2 :**

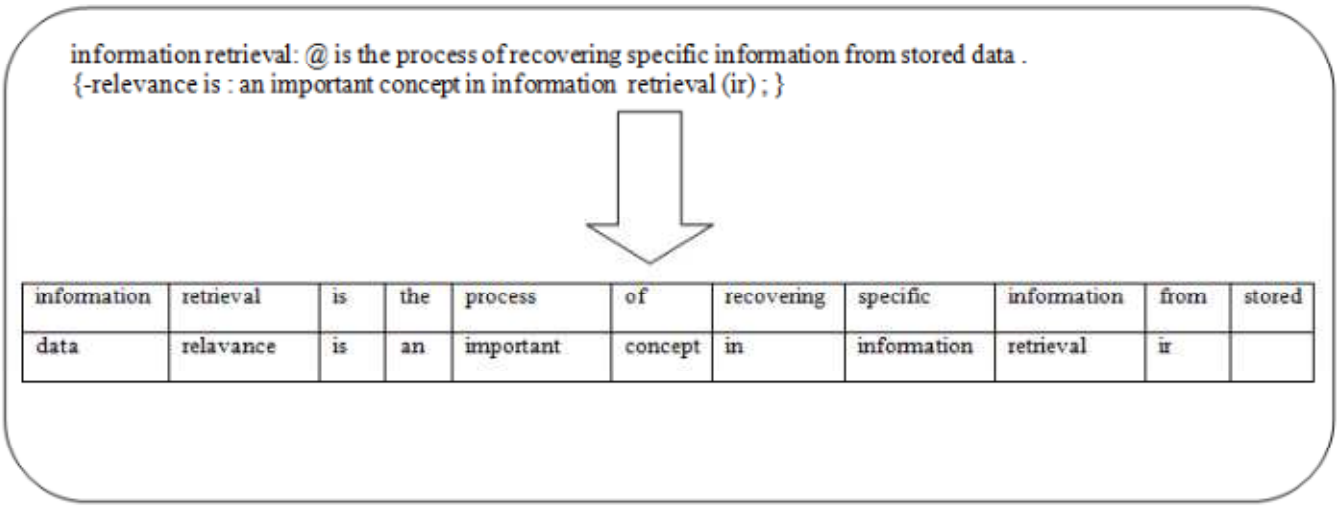



**Figure 2.2 segmentation de document**

### d. Elimination des mots vides :

Cette phase est très importante car elle réduit la taille d'un document par l'élimination des mots qui ont aucun intérêt dans l'étape de recherche, pour cela on a utilisé pour chaque langage un fichier des mots vides. Le tableau suivant illustre les différents fichiers des mots vides utilisés pour chaque langue.

| La langue | Le nombre des mots vides | Exemples |
|---|---|---|
| English | 635 | All, also, for, in |
| French | 463 | A, afin, entre, notre |
| Dutch | 259 | Alias, behalve, nogal, of |
| Finnish | 747 | Aiemin, jopa, kolme, yöskin |
| Germany | 129 | Aber, den, musst, sollen |
| Hungarian | 35 | Igen, ki, le, ti |
| Italian | 399 | Deve, dice, fuori, gia |
| Norwergian | 119 | Bruke, denne, kunne, lage |
| Portugese | 392 | Agora, breve, catorze, zero |
| Romanian | 281 | Acea, unuia, zece, treilea |
| Russian | 421 | А, е, и, ж |
| Spanish | 351 | Esta, actualmente, podemos |
| Swedish | 386 | Andra, hur, olikt |
| Turkish | 114 | Birþeyi, trilyon, çünkü |
| Danish | 64 | Når, og, endnu, hvad |

**Tableau 2.1 : Les mots vides des différentes langues.**

L'algorithme suivant illustre les étapes à suivre pour l'élimination des mots vides.

```
Entrée : - tableau de tokens .
         -listes des mots vides.
Sortie : tableau de tokens .
Début :
   Pour chaque token de tableau faire
   Si le token appartient a la liste des mots vides
      Alors  supprimer le token du tableau
   Finsi
  Finpour
Fin
```

L'application de l'élimination des mots vides sur l'exemple précédent est montrée dans **la figure 2.3** suivante :

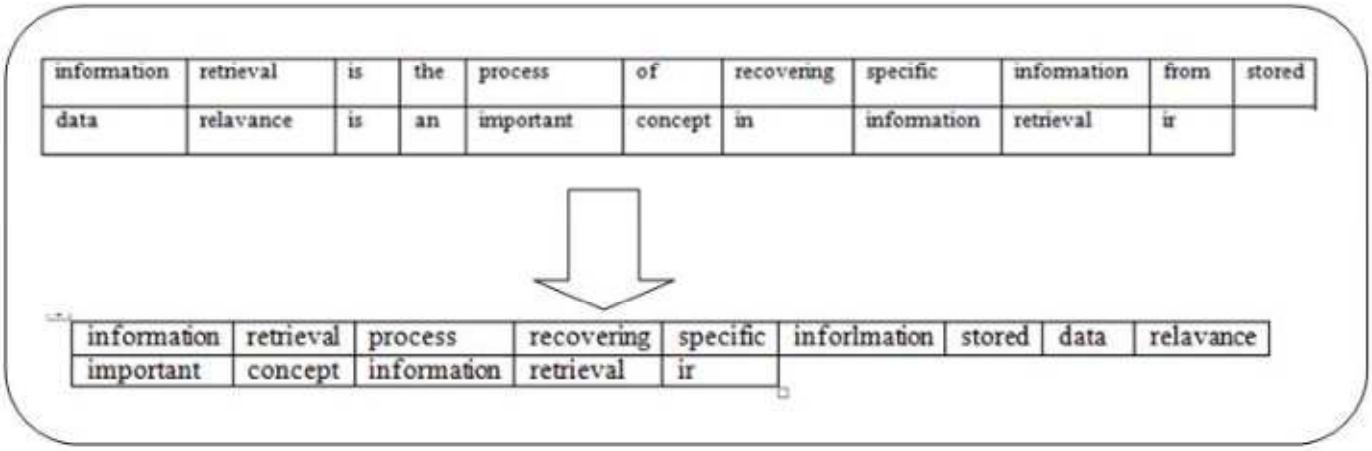



**Figure 2.3 Elimination des mots vides**

### e. Lemmatisation_:

La Lemmatisation permet de regrouper les mots de même famille, donc elle consiste à transformer un mot en une forme ''standard ''. Pour la langue anglaise par exemple, la lemmatisation des mots est réalisée avec la méthode de Porter (Porter, 1980). Pour trouver les lemmes des mots anglais, Porter fait juste des troncatures sur les terminaisons de mots

comme (suffixes, préfixes, post-fixe) pour trouver une forme standard. Par exemple le mot 'retrieval' est remplacé par son lemme 'retriev'.

L'algorithme suivant illustre la lemmatisation des mots :

```
Entrée : tableau de tokens.
Sortie : tableau de racines.
Début
Pour chaque token du tableau faire
    Racine ←porter (token)
    Insérer la racine dans le tableau de racines.
Finpour
Fin
```

L'application de la lemmatisation sur l'exemple précédant est indique dans **la figure 2.4** :

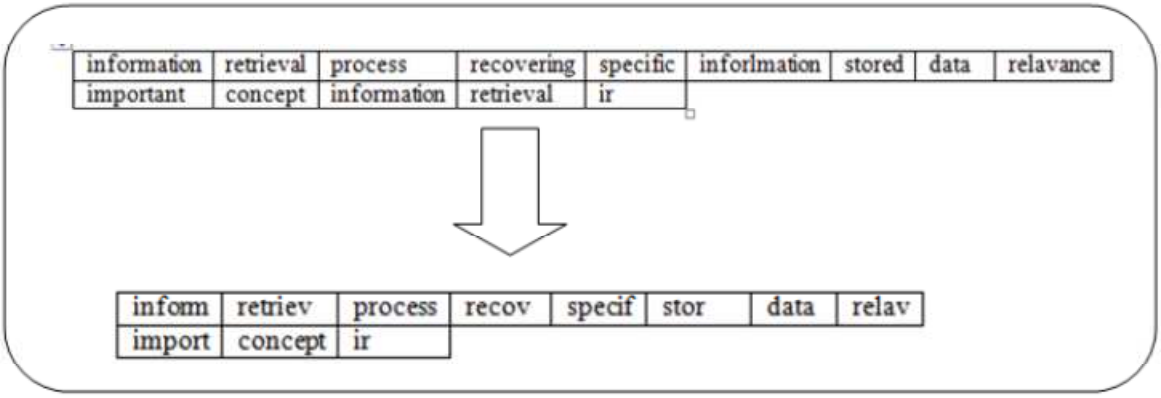



**Figure 2.4 lemmatisation des tokens**

## 2.2 Pondération :

La pondération consiste à calculer le poids d'un mot pour mesurer son importance dans le document, nous avons utilisé dans notre application trois types de fonction de pondération qui sont :

- Tf : leur avantage c'est que tout les fonctions de pondération manipule cette mesure, il indique le nombre d'occurrence de chaque mot dans chaque

document, telle que plus le mot est fréquent dans le document plus il est important dans ce document. Mais leur inconvénient que ne prend pas en considération la présence de ce mot dans les autres documents.

Par exemple on a deux mots 'mot1' et 'mot2'ont le même poids (TF =5) dans un document doc1, mais le mot2 existe dans plusieurs document par contre le mot1 se trouve que dans le document doc1, donc normalement le mot1 est plus important que le mot2.

La solution pour ce manque est la pondération **IDF**.

- Idf : c'est une représentation globale donc elle mesure l'importance d'un mot dans toute la collection,

c.à.d. un mot trop fréquent dans la collection est moins important qu'un mot moins fréquent.

- Tf*Idf : Celle ci combine les deux pondérations précédentes pour donner une meilleure représentation des mots dans les documents, une valeur de **TF* IDF** élevée pour un terme signifie que ce terme est important dans le document et en plus, qu'il apparait peu dans les autres documents de la collection. C'est le cas où un terme correspond à une caractéristique importante et unique dans un document.

Pour la pondération **IDF** le résultat est un tableau. Pour les autres pondérations, le résultat est une matrice où les lignes sont les documents de la collection et les colonnes sont les mots.

L'algorithme suivant illustre les étapes à suivre pour pondérer les mots :

Entrée : Tableau de token.

Sortie : Matrice (document×token).

Début

**Pour** chaque document i **faire**

**Pour** chaque mot j **faire**

Indice ← getindice (mot)

**Si** le token existe déjà dans le tableau **Alors**

Incrémenter la valeur de sa fréquence

Matrice[i][j]← Matrice[i][j]+1

**Sinon**

Stocker le mot dans le tableau et initialise sa fréquence à 1

Matrice[i][j] ← 1

Finpour

Finpour

Fin

La **figure 2.5** illustre un exemple de matrice de pondération**:**

| | $mot1$ | $mot2$ | $mot3$ | $mot4$ | . . . | $mot\ n$ |
|---|---|---|---|---|---|---|
| $doc\ 1$ | 3 | 2 | 5 | 1 | . . . | . |
| $doc\ 2$ | 5 | 4 | 2 | 0 | . . . | . |
| $doc\ 3$ | 0 | 1 | 0 | 0 | . . . | . |
| $doc\ 4$ | 1 | 0 | 0 | 2 | . . . | . |
| $doc\ 5$ | . | . | . | . | . . . | . |
| . | . | . | . | . | . . . | . |
| $doc\ n$ | . | . | . | . | . . . | . |

Tf (doc 1, mot1)=3 ⟶ le mot1 existe 3 fois dans le document 1.

Tf (doc 3, mot4)=0 ⟶ le document 3 ne contient pas le mot 4.

Idf mot1 =ln (4/ (3+1)) +1=1.

**Figure 2.5 : exemple d'une matrice de pondération**

## 2.3 Le prétraitement de la requête :

La requête exprimée par l'utilisateur sera indexé de la même façon que la collection. La seul différence réside dans l'étape de pondération, En effet, la pondération de la requête consiste à affecter le poids '1' pour les mots présent dans la requête et le poids '0' sinon.

L'algorithme suivant illustre la pondération de la requête :

```
Entrée : - la requête
         - tableau des tokens
Sortie : tableau des entiers « tab »
Début
Pour chaque mot de requête faire
Si le mot est trouvé dans le tableau des
tokens alors tab [mot] ← 1
Finsi
Finpour
Fin
```

### 2.4. L'appariement document requête :

La dernière étape consiste à comparer la requête avec les documents c.à.d. calculer un score de pertinence entre le vecteur requête et les vecteurs documents afin de mesurer le degré de rapproche entre ces deux représentations. Pour cela, on utilise une fonction de correspondance définit par des lois comme nous avons indiqué dans le chapitre précédent (tableau 1.2).

- ✓ Produit scalaire :

Utilisée pour mesurer le degré de pertinence entre un document et une requête, mais cette mesure ne donne pas un résultat efficace parce qu'il n'est pas normalisé.

$$RSV(q, di) = \sum_{j=1}^{|T|} w_{qj} . w_{ij}$$

- ✓ Cosinus

Le cosinus entre deux vecteurs est obtenu en calculant le produit scalaire entre ces deux vecteurs, que nous divisons par le produit de la norme des deux vecteurs donc cette mesure est normalisée et les valeurs calculée comprises entre '0' et '1' par la formule :

$$RSV(q, di) = \frac{\sum_{j=1}^{|T|} w_{qj} . w_{ij}}{\sqrt{\sum_{j=1}^{|T|} w_{qj}^{\ 2} \sum_{j=1}^{|T|} w_{ij}^{\ 2}}}$$

- ✓ Jaccard :

Le coefficient de Jaccard est défini comme étant le quotient du cardinal de l'intersection par celui de l'union, les valeurs calculée de cette mesure est entre '0' et '1' donc elle est normalisé, donner par :

$$RSV(q, di) = \frac{\sum_{j=1}^{|T|} w_{qj} . w_{ij}}{\sum_{j=1}^{|T|} w_{qj}^{\ 2} + \sum_{j=1}^{|T|} w_{ij}^{\ 2} - \sum_{j=1}^{|T|} w_{qj} . w_{ij}}$$

✓ Dice :

La similarité de *Dice* est définie par le nombre des objets communs multipliés par 2 sur le nombre total d'objets. La mesure de *Dice* est donc définie par la formule suivante :

$$RSV(q,d_i) = \frac{2*\sum_{j=1}^{|T|} w_{qj}.w_{ij}}{\sqrt{\sum_{j=1}^{|T|} w_{qj}^{\ 2} + \sum_{j=1}^{|T|} w_{ij}^{\ 2}}}$$

L'algorithme suivant illustre les étapes à suivre pour calculer le score entre le document et la requête :

Entrée : - le tableau de la requête

- une matrice

Sortie : tableau de score « rsv »

Début

**Pour** chaque document i **faire**

Calculer le score entre le document et la requête

rsv[i] ← la valeur de mesure calculée

Fin pour

Trier la table « rsv » par ordre décroissante et afficher les documents trie

Fin

Le résultat final de cette phase est un tableau des scores entre la requête et les documents. Dans le but de satisfaire la requête de l'utilisateur, il est nécessaire de trier le tableau des scores par ordre décroissant selon le degré de similarité afin de pouvoir restituer les documents les plus pertinents.

L'exemple de **la figure 2.6** illustre un exemple d'appariement entre la matrice de l'exemple précédent avec une requête Q : Q = [1 1 0 1]

$$\text{Matrice} = \begin{bmatrix} & mot1 & mot2 & mot3 & mot4 \\ doc\ 1 & 3 & 2 & 5 & 1 \\ doc\ 2 & 5 & 4 & 2 & 0 \\ doc\ 3 & 0 & 1 & 0 & 0 \\ doc\ 4 & 1 & 0 & 0 & 2 \end{bmatrix}$$

*Mesure de Dice entre la requête (q) et les documents de la matrice :

$$\text{Dice (q, doc)} = \frac{2*\sum_{j=1}^{M} Wqj.Wij}{\sum_{j=1}^{M} Wqj^2+\sum_{j=1}^{M} Wij^2}$$

$$\text{Dice(q,doc1)}=\frac{2*6}{(1^2+1^2+0^2+1^2)+(3^2+2^2+5^2+1^2)} = \frac{12}{42} = 0{,}28$$

$$\text{Dice(q,doc2)}=\frac{2*9}{(1^2+1^2+0^2+1^2)+(5^2+4^2+2^2+0^2)} = \frac{18}{48} = 0{,}37$$

$$\text{Dice(q,doc3)}=\frac{2*1}{(1^2+1^2+0^2+1^2)+(0^2+1^2+0^2+0^2)} = \frac{2}{4} = 0{,}5$$

$$\text{Dice(q,doc4)}=\frac{2*3}{(1^2+1^2+0^2+1^2)+(1^2+0^2+0^2+2^2)} = \frac{2}{8} = 0{,}75$$

**Figure 2.6 : Exemple de calcule de similarité par la mesure « Dice »**

D'après l'exemple, on peut déterminer que le document (doc4) est le document le plus pertinent pour la requête (q).

## 3. L'environnement de travail :

L'objectif principal de ce mémoire est de réaliser un système de recherche d'information, pour ce but nous avons choisi le langage de programmation « Java » avec l'Environnement de Développement Intégré (l'IDE) « NetBeans 8.0.2 ».

NetBeans IDE est l'IDE officielle de Java 8. Avec ses rédacteurs, analyseurs de code, et des convertisseurs, vous pouvez rapidement et facilement mettre à jour vos applications pour utiliser les nouvelles constructions Java 8 linguistiques, tels que lambdas, opérations fonctionnelles, et des références de méthode.

NetBeans IDE vous permet de développer Java Desktop rapidement et facilement, mobile, et les applications Web, ainsi que des applications HTML5 avec HTML, JavaScript et CSS. L'IDE fournit aussi un grand ensemble d'outils pour PHP et les développeurs C / C++. Il est gratuit et a une grande communauté d'utilisateurs et développeurs du monde entier.

NetBeans est disponible sous Windows, Mac, Linux et Solarisx86 et (SPARC) ou sous version indépendante du système d'exploitation, Un environnement Java Développement Kit JDK est requis pour les développements en Java.

## 4. Représentation de l'application :

➢ **Interface principale :**

La figure suivante représente l'interface principale de l'application.


**Figure 2.7 : L'interface principale de l'application**

➢ **La sélection de la langue :**

L'utilisateur doit sélectionner la langue qu'il veut, pour cela il doit choisir une langue dans le menu « langue ».

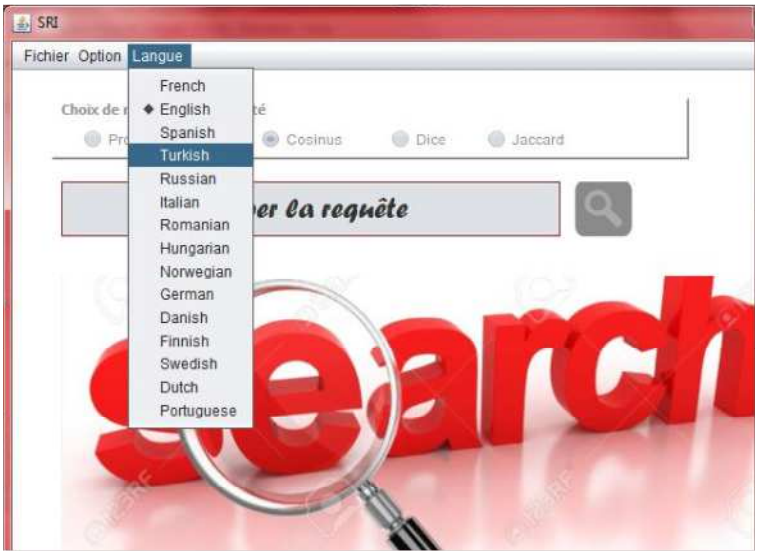



**Figure 2.8 : La sélection de langue de la recherche**

- **Indexation :**

➢ **Nouvelle indexation :**

Pour que le système puisse indexer le corpus, l'utilisateur doit choisir à travers le menu d'effectuer une nouvelle indexation (Fichier → Nouvelle indexation). La fenêtre indexation s'afficher comme indiqué dans la figure suivante :

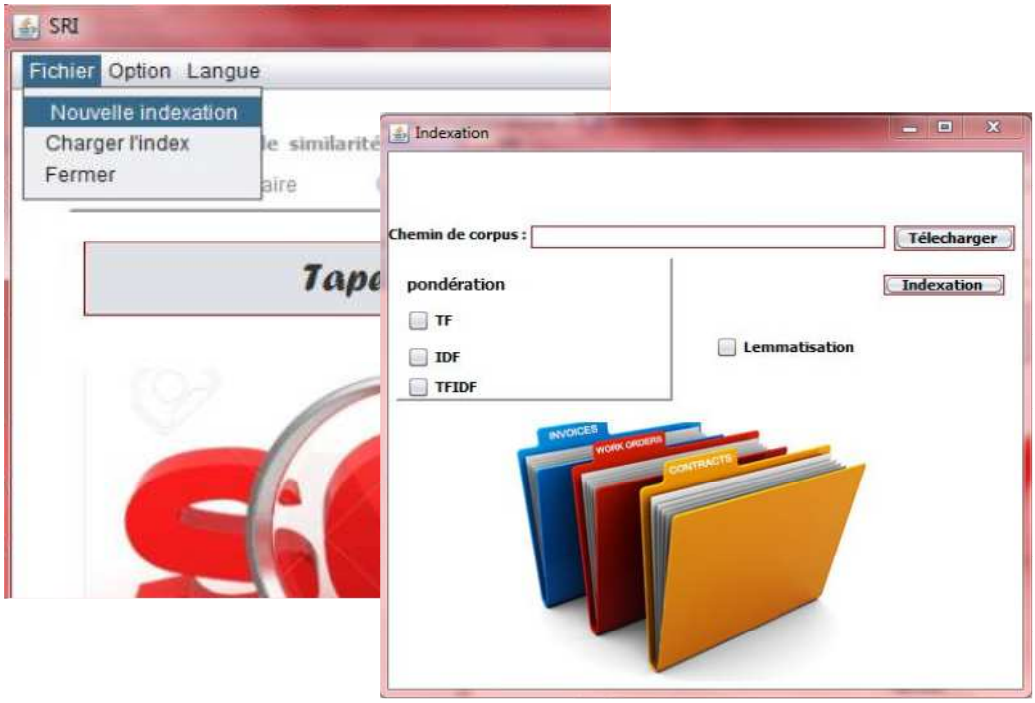



**Figure 2.9 : Les étapes d'indexation**

Dans cette interface, l'utilisateur doit choisir :

1- Le corpus en cliquant sur le bouton « télécharger »,
2- Une des méthodes de pondération,
3- La prise ou non prise en considération de la lemmatisation.

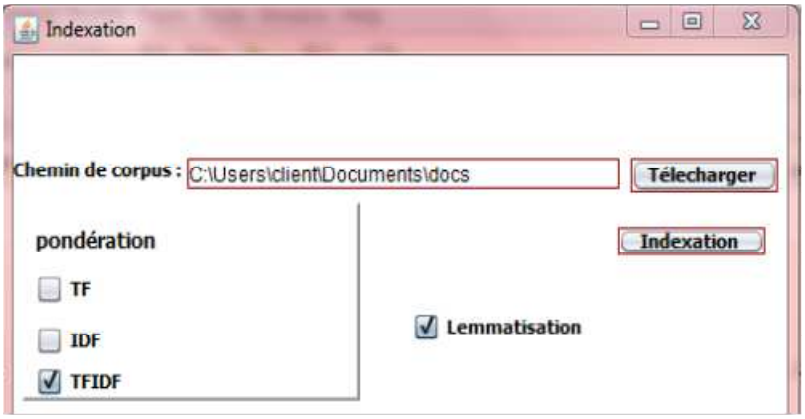



**Figure 2.10 : les propriétés d'indexation**

Une fois que l'utilisateur a initialisé ces choix, Il doit lancer l'indexation en cliquant sur le bouton « Indexation ».

Après avoir terminé l'indexation, notre système propose à l'utilisateur de sauvegarder l'indexation. Pour cela une autre interface s'affiche où l'utilisateur doit saisir le nom du fichier de sauvegarde.


**Figure 2.11 : L'enregistrement d'index**

- **Charger l'index :**

Si l'indexation est déjà effectuée et sauvegardée, l'utilisateur peut directement charger son index (Fichier -> Charger l'index).

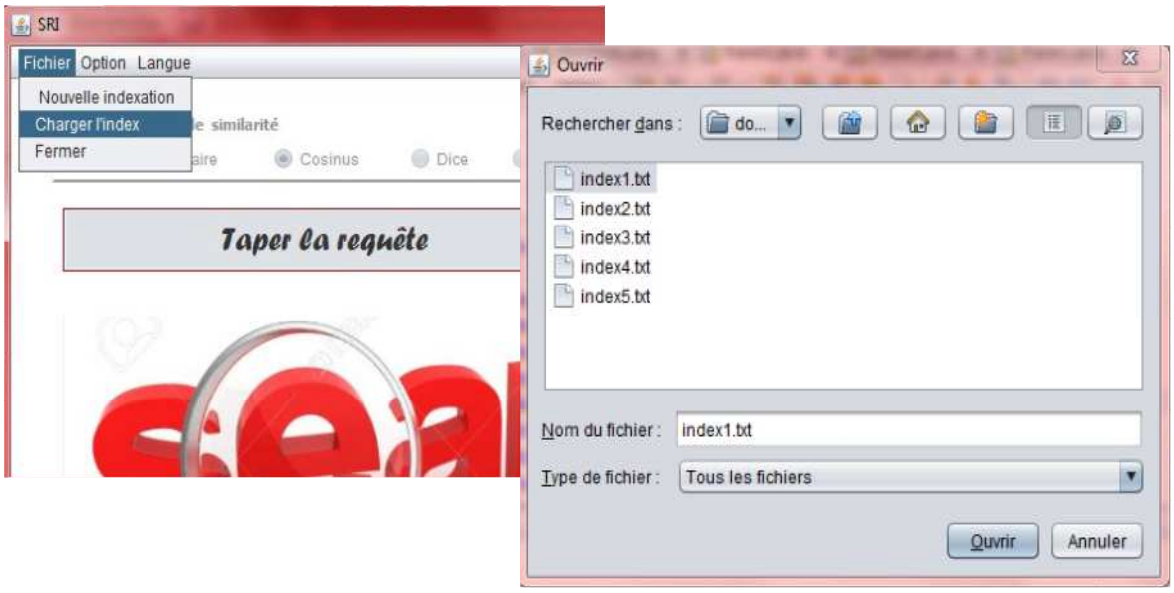



**Figure 2.12 : Le chargement d'index**

- **Appariement :**

Une fois l'indexation effectuée, l'utilisateur doit saisir sa requête et choisir une des mesures de similarité. La recherche est lancée en cliquant sur le bouton recherche

Le système retourne comme résultat l'ensemble des documents pertinents.

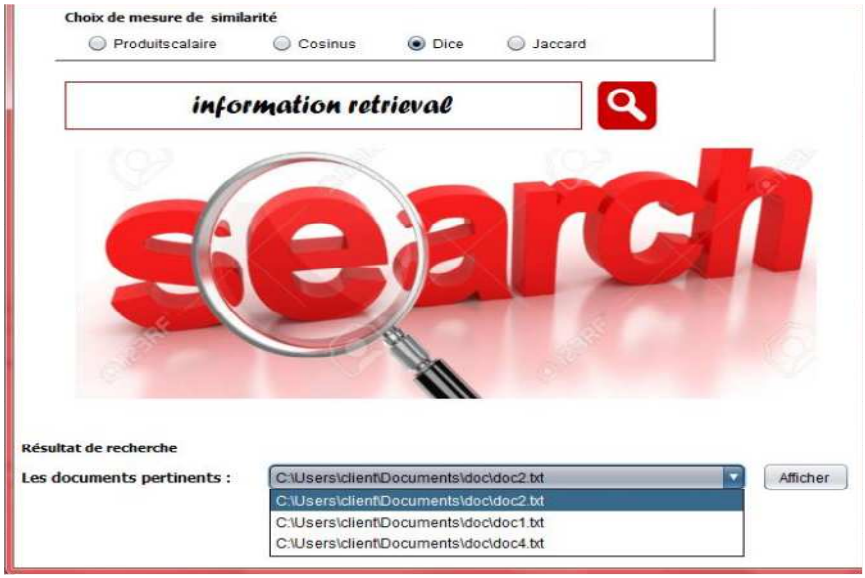



**Figure 2.13 : Le résultat de la recherche**

Dans le cas où le système ne trouve aucun résultat, l'interface suivant s'affiché :


**Figure 2.14 : Aucun document pertinent**

- **Visualisation des résultats :**

Notre système donne la possibilité à l'utilisateur :

1- d'afficher le contenu des documents ainsi que leurs scores en sélectionnant un document à partir de la liste des documents retournés. Pour cela, l'utilisateur doit cliquer sur le bouton « Afficher ».


**Figure 2.15 : Affichage de document sélectionné**

2- Afficher le dictionnaire :

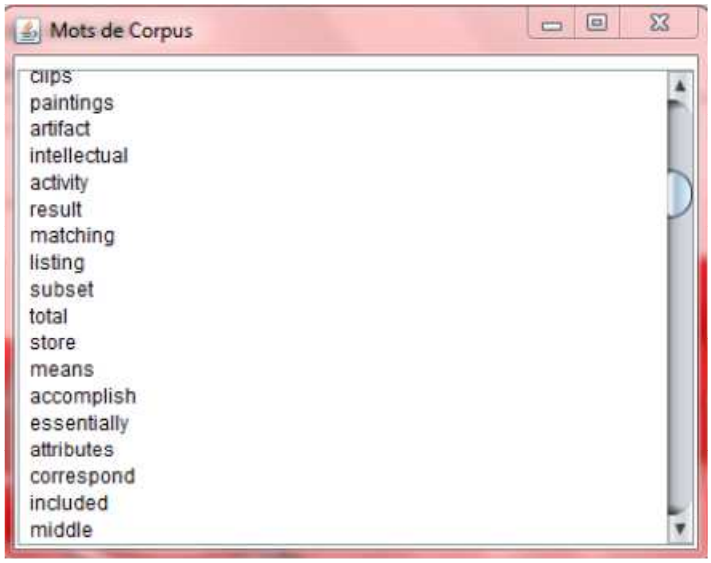



**Figure 2.16 : L'ensemble des mots du corpus.**

1- La matrice d'indexation (Option -> Résultat d'indexation) :

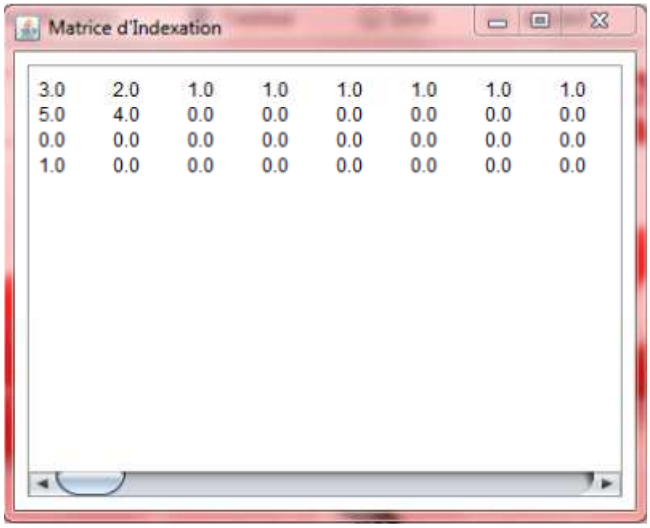



**Figure 2.17 : Résultat de la matrice**

2- Mesure de similarité (Option -> résultat de mesures) :

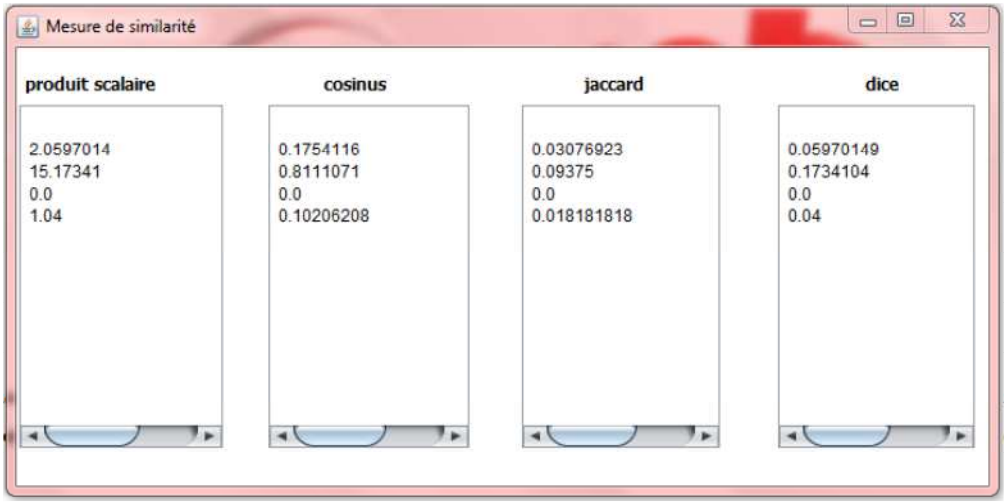


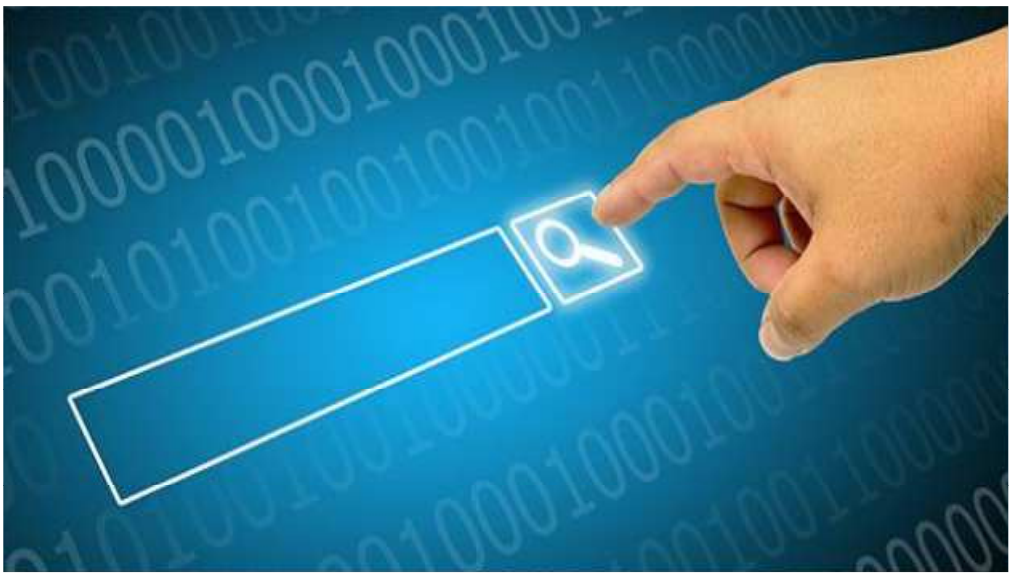

**Figure 2.18 : Résultats des mesures de similarité.**

## 5. Conclusion :

Dans ce chapitre nous avons présenté l'implémentation de notre application, nous avons montré en détail les différentes étapes nécessaires pour le bon fonctionnement d'un SRI qui répond précisément au besoin informationnel de l'utilisateur.


# Chapitre II

# Réalisation de l'Application



## 1. Introduction :

Un SRI manipule un corpus de documents qu'il transpose à l'aide d'une fonction d'indexation en un corpus indexé. Ce corpus lui permet de résoudre des requêtes traduites à partir de besoins utilisateur.

Un tel système repose sur la définition d'un modèle de RI qui effectue ces deux transpositions et qui fait correspondre les documents aux requêtes par une relation de pertinence.

Après avoir donné un aperçu générale de la RI, nous passons à l'implémentation de l'application. Dans ce chapitre on va se baser sur le modèle le plus utilisée qui est le modèle vectoriel pour la réalisation de notre SRI. Plus précisément, nous allons détailler les étapes suivies pour la réalisation de notre SRI en montrant les différentes techniques utilisées pour la pondération ainsi que les mesures de similarité utilisés.

## 2. Les étapes de la réalisation du système :

### 2.1 Le prétraitement :

Le prétraitement constitue la première étape dans la réalisation où on va indexer et analyser chaque document dans le corpus :

#### a. L'ouverture et la lecture de document texte :

On va ouvrir chaque fichier sous forme d'une chaine de caractère pour pouvoir appliquer les différents traitements, voici un exemple de la lecture d'un fichier :

```
INFORMATION RETRIEVAL: @
Is the process of recovering specific information from stored data .
{-Relevance Is : an important concept in information  retrieval (IR) ; }
```

#### b. Formater les mots en minuscule :

Pour que le système soit capable de connaitre tous les mots, il est nécessaire d'éliminer les majuscules, car un mot écrit en majuscule est considéré comme étant différent lorsqu'il est écrit en minuscule.

information retrieval : @
is the process of recovering specific information from stored data .
{-relevance is : an important concept in information retrieval (ir) ; }

### c. Segmentation des documents :

Dans cette étape, il s'agit d'extraire les mots contenus dans chaque fichier. Dans notre système un mot est considéré comme une suite de caractères délimités entre deux séparateurs. L'ensemble des séparateurs utilisé est montré dans la **figure 2.1** :

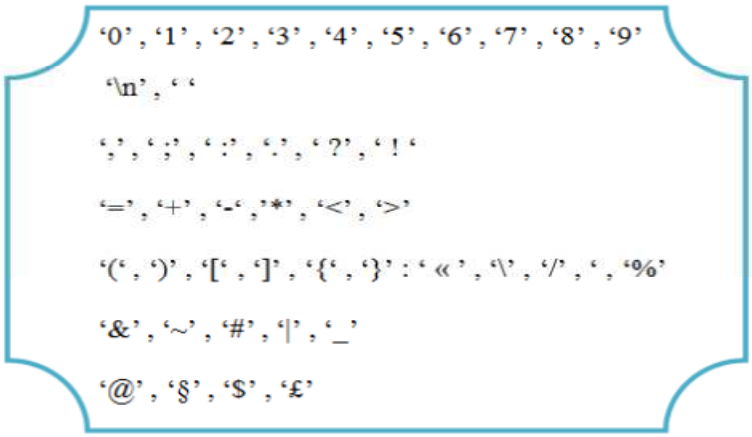



**Figure 2.1 : les séparateurs**

L'algorithme suivant illustre les étapes à suivre pour la segmentation de corpus.

```
Entrée : - corpus.
         -liste de séparateurs (délimiter).
Sortie : tableau de tokens.
Variable : chaine de caractère : ligne ;  tableau : tab
Début :
Ouvrir le corpus
Pour chaque document de corpus faire
    Line ← lire ligne
    Tant que line lu est différent EOF faire
        Tokenizer (line, délimiter)
            Tant que  il y'a des tokens  faire
            tokens ←  tokens suivant
            insérer le tokens dans  tab
            Fin tantque
            line ← lire ligne
    Fin tantque
  document suivant
 Finpour
Fin .
```

Après l'élimination des séparateurs, les mots (tokens) sont représentés comme le montre **la figure 2.2 :**

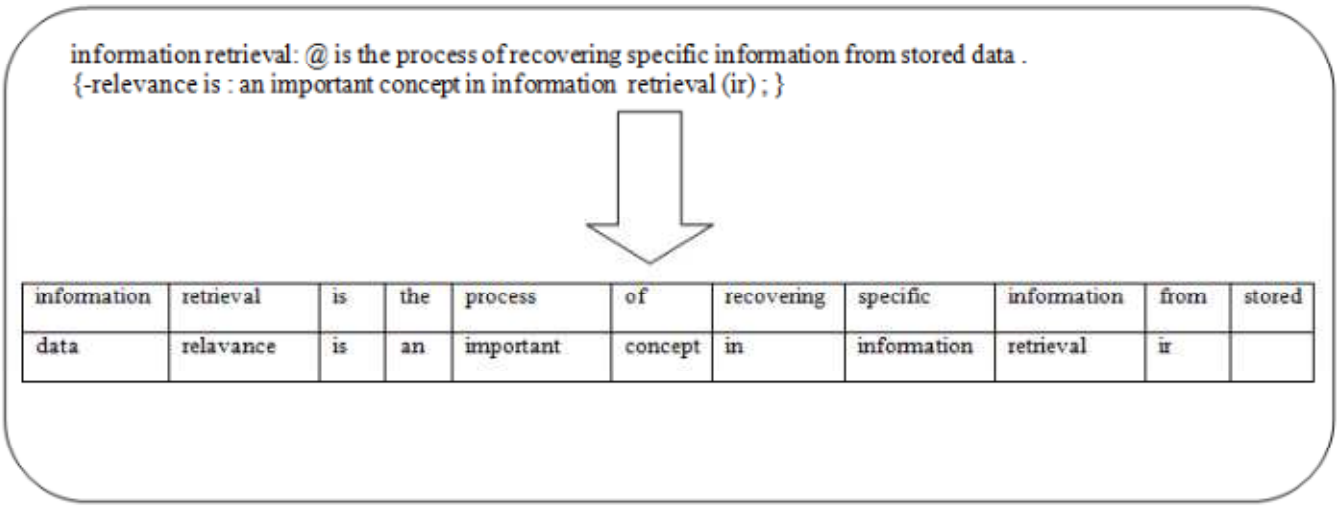



**Figure 2.2 segmentation de document**

### d. Elimination des mots vides :

Cette phase est très importante car elle réduit la taille d'un document par l'élimination des mots qui ont aucun intérêt dans l'étape de recherche, pour cela on a utilisé pour chaque langage un fichier des mots vides. Le tableau suivant illustre les différents fichiers des mots vides utilisés pour chaque langue.

| La langue | Le nombre des mots vides | Exemples |
|---|---|---|
| English | 635 | All, also, for, in |
| French | 463 | A, afin, entre, notre |
| Dutch | 259 | Alias, behalve, nogal, of |
| Finnish | 747 | Aiemin, jopa, kolme, yöskin |
| Germany | 129 | Aber, den, musst, sollen |
| Hungarian | 35 | Igen, ki, le, ti |
| Italian | 399 | Deve, dice, fuori, gia |
| Norwergian | 119 | Bruke, denne, kunne, lage |
| Portugese | 392 | Agora, breve, catorze, zero |
| Romanian | 281 | Acea, unuia, zece, treilea |
| Russian | 421 | А, е, и, ж |
| Spanish | 351 | Esta, actualmente, podemos |
| Swedish | 386 | Andra, hur, olikt |
| Turkish | 114 | Birþeyi, trilyon, çünkü |
| Danish | 64 | Når, og, endnu, hvad |

**Tableau 2.1 : Les mots vides des différentes langues.**

L'algorithme suivant illustre les étapes à suivre pour l'élimination des mots vides.

```
Entrée : - tableau de tokens .
         -listes des mots vides.
Sortie : tableau de tokens .
Début :
   Pour chaque token de tableau faire
   Si le token appartient a la liste des mots vides
      Alors  supprimer le token du tableau
   Finsi
  Finpour
Fin
```

L'application de l'élimination des mots vides sur l'exemple précédent est montrée dans **la figure 2.3** suivante :

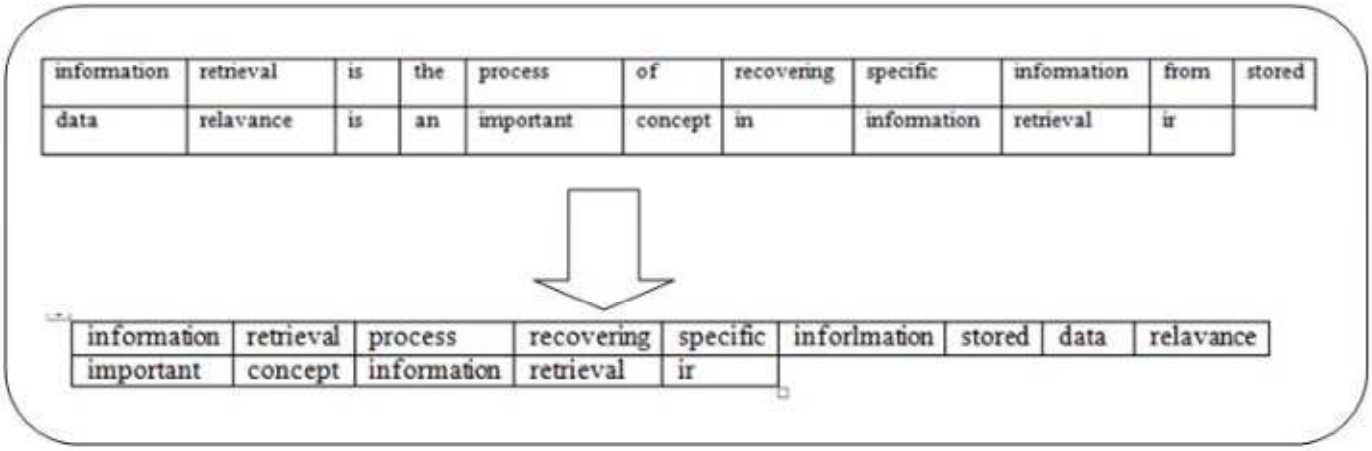



**Figure 2.3 Elimination des mots vides**

### e. Lemmatisation_:

La Lemmatisation permet de regrouper les mots de même famille, donc elle consiste à transformer un mot en une forme ''standard ''. Pour la langue anglaise par exemple, la lemmatisation des mots est réalisée avec la méthode de Porter (Porter, 1980). Pour trouver les lemmes des mots anglais, Porter fait juste des troncatures sur les terminaisons de mots

comme (suffixes, préfixes, post-fixe) pour trouver une forme standard. Par exemple le mot 'retrieval' est remplacé par son lemme 'retriev'.

L'algorithme suivant illustre la lemmatisation des mots :

```
Entrée : tableau de tokens.
Sortie : tableau de racines.
Début
Pour chaque token du tableau faire
    Racine ←porter (token)
    Insérer la racine dans le tableau de racines.
Finpour
Fin
```

L'application de la lemmatisation sur l'exemple précédant est indique dans **la figure 2.4** :

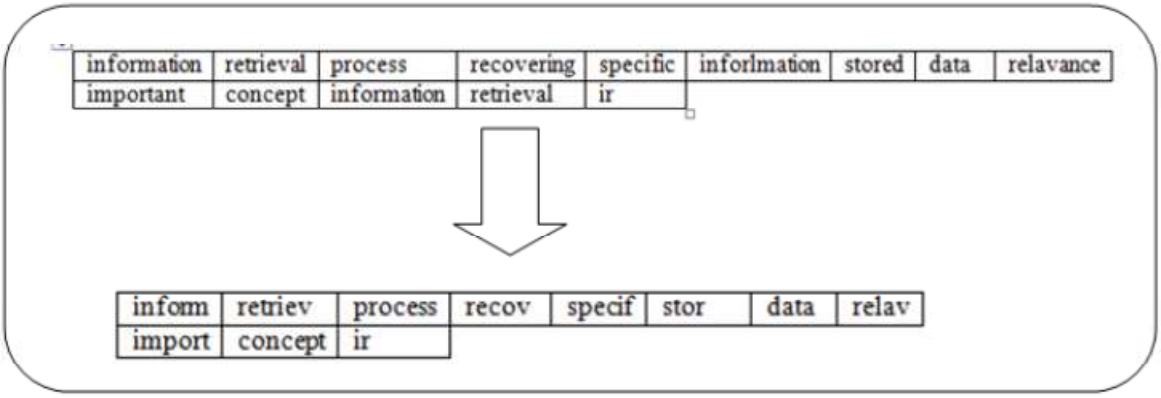



**Figure 2.4 lemmatisation des tokens**

## 2.2 Pondération :

La pondération consiste à calculer le poids d'un mot pour mesurer son importance dans le document, nous avons utilisé dans notre application trois types de fonction de pondération qui sont :

- Tf : leur avantage c'est que tout les fonctions de pondération manipule cette mesure, il indique le nombre d'occurrence de chaque mot dans chaque

document, telle que plus le mot est fréquent dans le document plus il est important dans ce document. Mais leur inconvénient que ne prend pas en considération la présence de ce mot dans les autres documents.

Par exemple on a deux mots 'mot1' et 'mot2'ont le même poids (TF =5) dans un document doc1, mais le mot2 existe dans plusieurs document par contre le mot1 se trouve que dans le document doc1, donc normalement le mot1 est plus important que le mot2.

La solution pour ce manque est la pondération **IDF**.

- Idf : c'est une représentation globale donc elle mesure l'importance d'un mot dans toute la collection,

c.à.d. un mot trop fréquent dans la collection est moins important qu'un mot moins fréquent.

- Tf*Idf : Celle ci combine les deux pondérations précédentes pour donner une meilleure représentation des mots dans les documents, une valeur de **TF* IDF** élevée pour un terme signifie que ce terme est important dans le document et en plus, qu'il apparait peu dans les autres documents de la collection. C'est le cas où un terme correspond à une caractéristique importante et unique dans un document.

Pour la pondération **IDF** le résultat est un tableau. Pour les autres pondérations, le résultat est une matrice où les lignes sont les documents de la collection et les colonnes sont les mots.

L'algorithme suivant illustre les étapes à suivre pour pondérer les mots :

Entrée : Tableau de token.

Sortie : Matrice (document×token).

Début

**Pour** chaque document i **faire**

**Pour** chaque mot j **faire**

Indice ← getindice (mot)

**Si** le token existe déjà dans le tableau **Alors**

Incrémenter la valeur de sa fréquence

Matrice[i][j]← Matrice[i][j]+1

**Sinon**

Stocker le mot dans le tableau et initialise sa fréquence à 1

Matrice[i][j] ← 1

Finpour

Finpour

Fin

La **figure 2.5** illustre un exemple de matrice de pondération**:**

| | $mot1$ | $mot2$ | $mot3$ | $mot4$ | . . . | $mot\ n$ |
|---|---|---|---|---|---|---|
| $doc\ 1$ | 3 | 2 | 5 | 1 | . . . | . |
| $doc\ 2$ | 5 | 4 | 2 | 0 | . . . | . |
| $doc\ 3$ | 0 | 1 | 0 | 0 | . . . | . |
| $doc\ 4$ | 1 | 0 | 0 | 2 | . . . | . |
| $doc\ 5$ | . | . | . | . | . . . | . |
| . | . | . | . | . | . . . | . |
| $doc\ n$ | . | . | . | . | . . . | . |

Tf (doc 1, mot1)=3 ⟶ le mot1 existe 3 fois dans le document 1.

Tf (doc 3, mot4)=0 ⟶ le document 3 ne contient pas le mot 4.

Idf mot1 =ln (4/ (3+1)) +1=1.

**Figure 2.5 : exemple d'une matrice de pondération**

## 2.3 Le prétraitement de la requête :

La requête exprimée par l'utilisateur sera indexé de la même façon que la collection. La seul différence réside dans l'étape de pondération, En effet, la pondération de la requête consiste à affecter le poids '1' pour les mots présent dans la requête et le poids '0' sinon.

L'algorithme suivant illustre la pondération de la requête :

```
Entrée : - la requête
         - tableau des tokens
Sortie : tableau des entiers « tab »
Début
Pour chaque mot de requête faire
Si le mot est trouvé dans le tableau des
tokens alors tab [mot] ← 1
Finsi
Finpour
Fin
```

### 2.4. L'appariement document requête :

La dernière étape consiste à comparer la requête avec les documents c.à.d. calculer un score de pertinence entre le vecteur requête et les vecteurs documents afin de mesurer le degré de rapproche entre ces deux représentations. Pour cela, on utilise une fonction de correspondance définit par des lois comme nous avons indiqué dans le chapitre précédent (tableau 1.2).

- ✓ Produit scalaire :

Utilisée pour mesurer le degré de pertinence entre un document et une requête, mais cette mesure ne donne pas un résultat efficace parce qu'il n'est pas normalisé.

$$RSV(q, di) = \sum_{j=1}^{|T|} w_{qj} . w_{ij}$$

- ✓ Cosinus

Le cosinus entre deux vecteurs est obtenu en calculant le produit scalaire entre ces deux vecteurs, que nous divisons par le produit de la norme des deux vecteurs donc cette mesure est normalisée et les valeurs calculée comprises entre '0' et '1' par la formule :

$$RSV(q, di) = \frac{\sum_{j=1}^{|T|} w_{qj} . w_{ij}}{\sqrt{\sum_{j=1}^{|T|} w_{qj}^{\ 2} \sum_{j=1}^{|T|} w_{ij}^{\ 2}}}$$

- ✓ Jaccard :

Le coefficient de Jaccard est défini comme étant le quotient du cardinal de l'intersection par celui de l'union, les valeurs calculée de cette mesure est entre '0' et '1' donc elle est normalisé, donner par :

$$RSV(q, di) = \frac{\sum_{j=1}^{|T|} w_{qj} . w_{ij}}{\sum_{j=1}^{|T|} w_{qj}^{\ 2} + \sum_{j=1}^{|T|} w_{ij}^{\ 2} - \sum_{j=1}^{|T|} w_{qj} . w_{ij}}$$

✓ Dice :

La similarité de *Dice* est définie par le nombre des objets communs multipliés par 2 sur le nombre total d'objets. La mesure de *Dice* est donc définie par la formule suivante :

$$RSV(q,d_i) = \frac{2*\sum_{j=1}^{|T|} w_{qj}.w_{ij}}{\sqrt{\sum_{j=1}^{|T|} w_{qj}^{\ 2} + \sum_{j=1}^{|T|} w_{ij}^{\ 2}}}$$

L'algorithme suivant illustre les étapes à suivre pour calculer le score entre le document et la requête :

Entrée : - le tableau de la requête

- une matrice

Sortie : tableau de score « rsv »

Début

**Pour** chaque document i **faire**

Calculer le score entre le document et la requête

rsv[i] ← la valeur de mesure calculée

Fin pour

Trier la table « rsv » par ordre décroissante et afficher les documents trie

Fin

Le résultat final de cette phase est un tableau des scores entre la requête et les documents. Dans le but de satisfaire la requête de l'utilisateur, il est nécessaire de trier le tableau des scores par ordre décroissant selon le degré de similarité afin de pouvoir restituer les documents les plus pertinents.

L'exemple de **la figure 2.6** illustre un exemple d'appariement entre la matrice de l'exemple précédent avec une requête Q : Q = [1 1 0 1]

$$\text{Matrice} = \begin{bmatrix} & mot1 & mot2 & mot3 & mot4 \\ doc\ 1 & 3 & 2 & 5 & 1 \\ doc\ 2 & 5 & 4 & 2 & 0 \\ doc\ 3 & 0 & 1 & 0 & 0 \\ doc\ 4 & 1 & 0 & 0 & 2 \end{bmatrix}$$

*Mesure de Dice entre la requête (q) et les documents de la matrice :

$$\text{Dice (q, doc)} = \frac{2*\sum_{j=1}^{M} Wqj.Wij}{\sum_{j=1}^{M} Wqj^2+\sum_{j=1}^{M} Wij^2}$$

$$\text{Dice(q,doc1)}=\frac{2*6}{(1^2+1^2+0^2+1^2)+(3^2+2^2+5^2+1^2)} = \frac{12}{42} = 0{,}28$$

$$\text{Dice(q,doc2)}=\frac{2*9}{(1^2+1^2+0^2+1^2)+(5^2+4^2+2^2+0^2)} = \frac{18}{48} = 0{,}37$$

$$\text{Dice(q,doc3)}=\frac{2*1}{(1^2+1^2+0^2+1^2)+(0^2+1^2+0^2+0^2)} = \frac{2}{4} = 0{,}5$$

$$\text{Dice(q,doc4)}=\frac{2*3}{(1^2+1^2+0^2+1^2)+(1^2+0^2+0^2+2^2)} = \frac{2}{8} = 0{,}75$$

**Figure 2.6 : Exemple de calcule de similarité par la mesure « Dice »**

D'après l'exemple, on peut déterminer que le document (doc4) est le document le plus pertinent pour la requête (q).

## 3. L'environnement de travail :

L'objectif principal de ce mémoire est de réaliser un système de recherche d'information, pour ce but nous avons choisi le langage de programmation « Java » avec l'Environnement de Développement Intégré (l'IDE) « NetBeans 8.0.2 ».

NetBeans IDE est l'IDE officielle de Java 8. Avec ses rédacteurs, analyseurs de code, et des convertisseurs, vous pouvez rapidement et facilement mettre à jour vos applications pour utiliser les nouvelles constructions Java 8 linguistiques, tels que lambdas, opérations fonctionnelles, et des références de méthode.

NetBeans IDE vous permet de développer Java Desktop rapidement et facilement, mobile, et les applications Web, ainsi que des applications HTML5 avec HTML, JavaScript et CSS. L'IDE fournit aussi un grand ensemble d'outils pour PHP et les développeurs C / C++. Il est gratuit et a une grande communauté d'utilisateurs et développeurs du monde entier.

NetBeans est disponible sous Windows, Mac, Linux et Solarisx86 et (SPARC) ou sous version indépendante du système d'exploitation, Un environnement Java Développement Kit JDK est requis pour les développements en Java.

## 4. Représentation de l'application :

➢ **Interface principale :**

La figure suivante représente l'interface principale de l'application.


**Figure 2.7 : L'interface principale de l'application**

➢ **La sélection de la langue :**

L'utilisateur doit sélectionner la langue qu'il veut, pour cela il doit choisir une langue dans le menu « langue ».

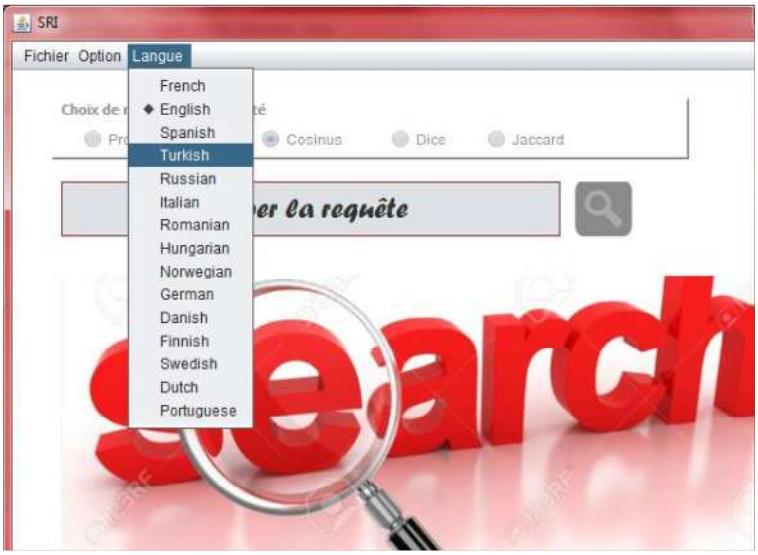



**Figure 2.8 : La sélection de langue de la recherche**

- **Indexation :**

➢ **Nouvelle indexation :**

Pour que le système puisse indexer le corpus, l'utilisateur doit choisir à travers le menu d'effectuer une nouvelle indexation (Fichier → Nouvelle indexation). La fenêtre indexation s'afficher comme indiqué dans la figure suivante :

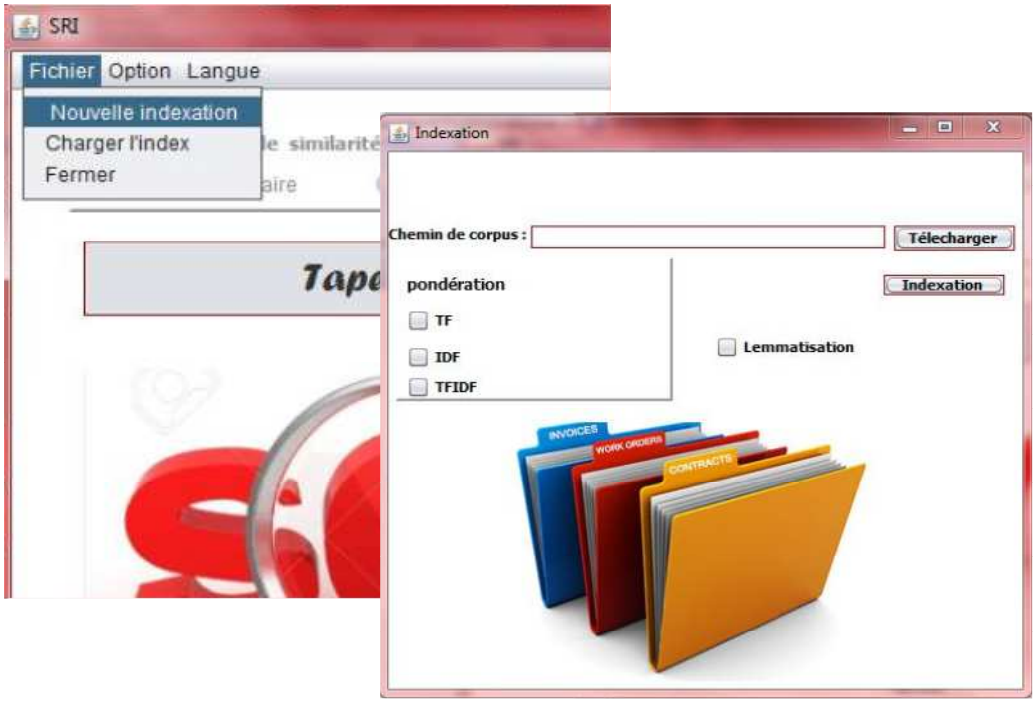



**Figure 2.9 : Les étapes d'indexation**

Dans cette interface, l'utilisateur doit choisir :

1- Le corpus en cliquant sur le bouton « télécharger »,
2- Une des méthodes de pondération,
3- La prise ou non prise en considération de la lemmatisation.

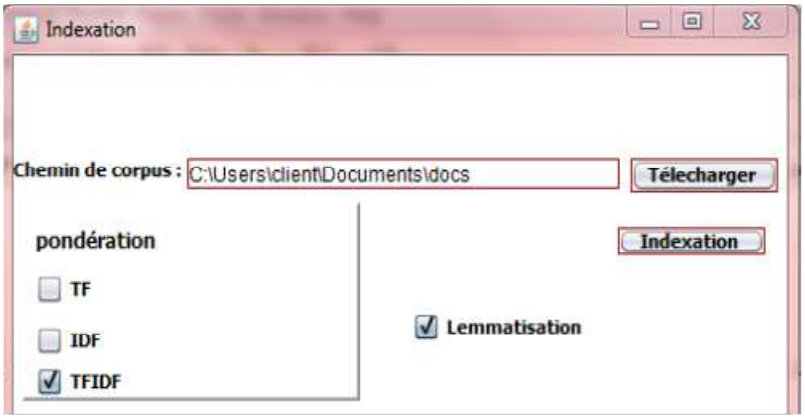



**Figure 2.10 : les propriétés d'indexation**

Une fois que l'utilisateur a initialisé ces choix, Il doit lancer l'indexation en cliquant sur le bouton « Indexation ».

Après avoir terminé l'indexation, notre système propose à l'utilisateur de sauvegarder l'indexation. Pour cela une autre interface s'affiche où l'utilisateur doit saisir le nom du fichier de sauvegarde.


**Figure 2.11 : L'enregistrement d'index**

- **Charger l'index :**

Si l'indexation est déjà effectuée et sauvegardée, l'utilisateur peut directement charger son index (Fichier -> Charger l'index).

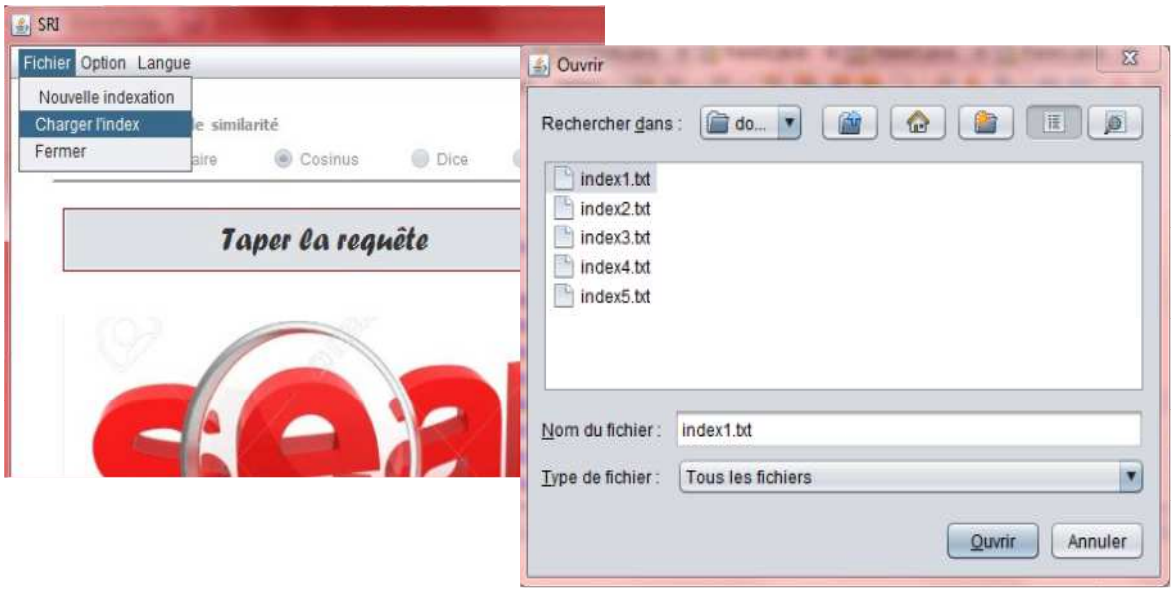



**Figure 2.12 : Le chargement d'index**

- **Appariement :**

Une fois l'indexation effectuée, l'utilisateur doit saisir sa requête et choisir une des mesures de similarité. La recherche est lancée en cliquant sur le bouton recherche

Le système retourne comme résultat l'ensemble des documents pertinents.

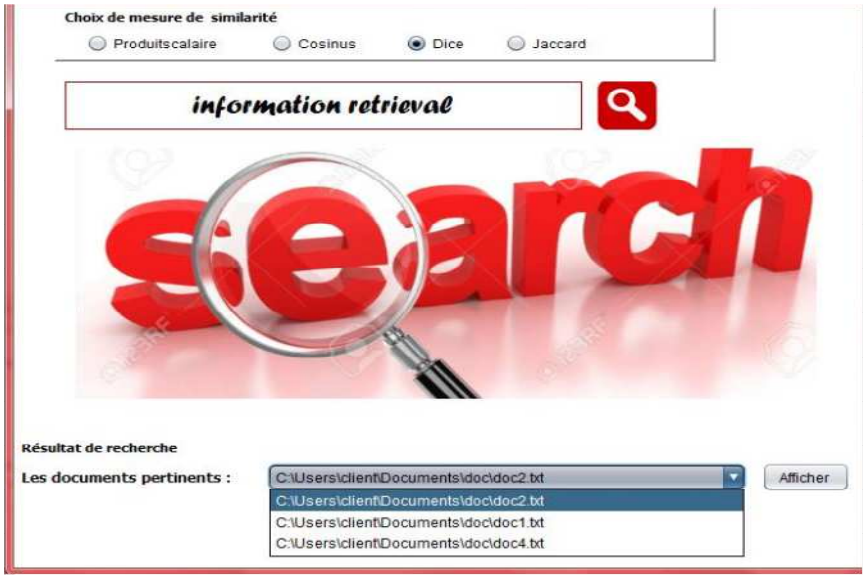



**Figure 2.13 : Le résultat de la recherche**

Dans le cas où le système ne trouve aucun résultat, l'interface suivant s'affiché :


**Figure 2.14 : Aucun document pertinent**

- **Visualisation des résultats :**

Notre système donne la possibilité à l'utilisateur :

1- d'afficher le contenu des documents ainsi que leurs scores en sélectionnant un document à partir de la liste des documents retournés. Pour cela, l'utilisateur doit cliquer sur le bouton « Afficher ».


**Figure 2.15 : Affichage de document sélectionné**

2- Afficher le dictionnaire :

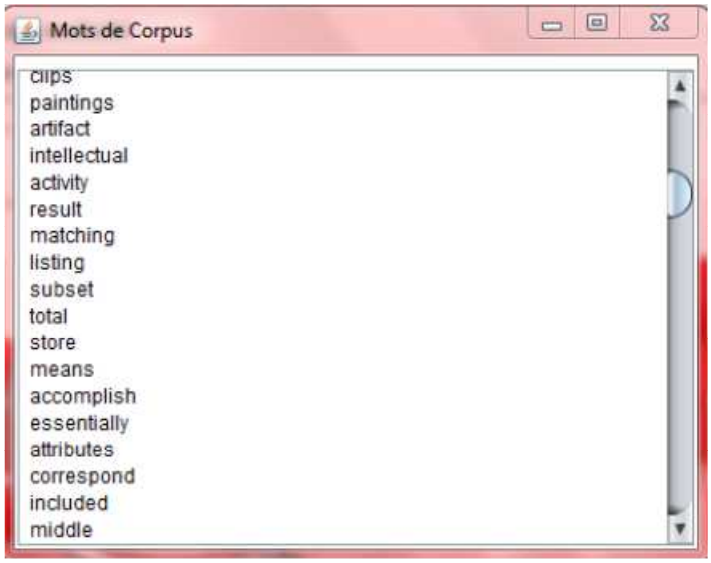



**Figure 2.16 : L'ensemble des mots du corpus.**

1- La matrice d'indexation (Option -> Résultat d'indexation) :

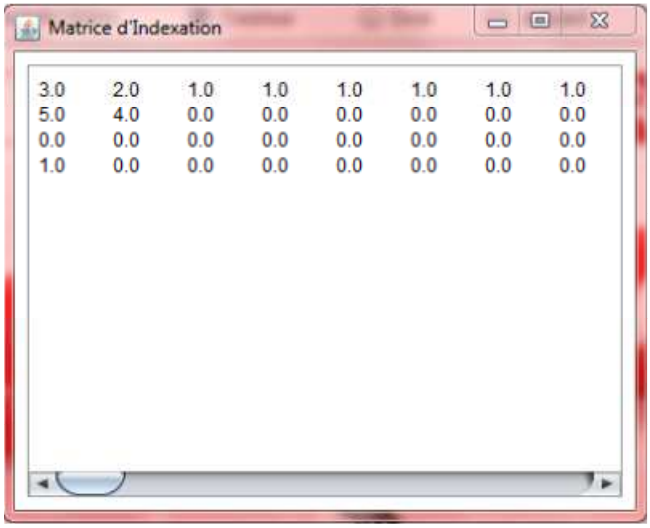



**Figure 2.17 : Résultat de la matrice**

2- Mesure de similarité (Option -> résultat de mesures) :

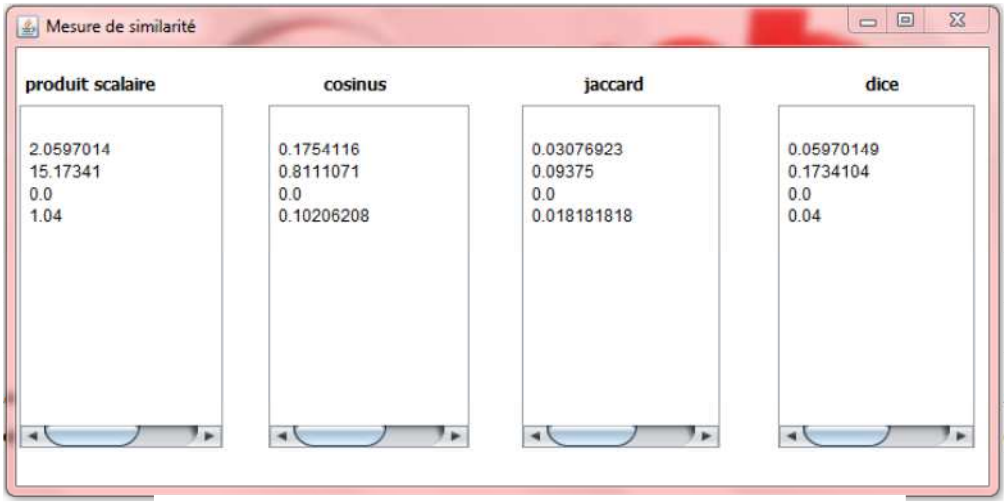



**Figure 2.18 : Résultats des mesures de similarité.**

## 5. Conclusion :

Dans ce chapitre nous avons présenté l'implémentation de notre application, nous avons montré en détail les différentes étapes nécessaires pour le bon fonctionnement d'un SRI qui répond précisément au besoin informationnel de l'utilisateur.

# Conclusion générale

Les travaux présentés dans ce mémoire rentrent dans le cadre de la conception et la réalisation d'un système de recherche d'information SRI.

Nous concluons que la recherche d'information, s'attache à définir des modèles de RI et des systèmes (SRI) , afin de faciliter l'accès à un ensemble de documents se trouvant dans des bases documentaires .

Le but d'un SRI 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 des 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.

Dans ce mémoire, nous avons présentées un bref historique d'IR. puis ce que peut être un SRI et ses différents modèles, ainsi nous avons rappelé ce qu'est l'indexation , ses étapes utilisée et ses techniques de pondération dont les formules améliorent de manière significative les résultats de recherche, comparativement à l'un des modèles de RI, on disperse les termes dans les documents pour un meilleur appariement requête-document.

Nous avons utilisé aussi quelques fonctions de similarité qui mesurent le degré proximité entre un document et une requête, en vue d'ordonner les documents les plus similaires à la requête au moins similaires. Enfin, nous présentons l'essentiel de notre travail et les étapes importantes de notre application.

**[01]** A. BOURAMOUL,“ Recherche d’information contextuelle et sémantique ”, thèse de doctorat en informatique, Université Mentouri de Constantine, (2011).

**[02]** G. SALTON and M.J. McGill,’’ Introduction to modern information retrieval’’, McGraw Hill Publishing Company, New York, (1983).

**[03]** H .TEBRI,’’ Formalisation et spécification d’un système de filtrage incrémental d’information’’, Thèse de doctorat de l’université Paul Sabatier, Toulouse, (2004).

**[04]** N. SMAIL,’’ Contribution a l'analyse et a la recherche d'information en texte intégral : application de la transformée en ondelettes pour la recherche et l'analyse de textes’’, (2010).

**[05]** F. DAHAK,’’Indexation des documents Semi-structurés’’, Mémoire de Magister, (2005).

**[06]** M. BAZIZ,’’Indexation conceptuelle guidée Par ontologie pour la recherche d’information’’, thèse doctorat de l’université de Paul Sabatier, (2005).

**[07]** H.BRINI, “Un model de recherche d’information basé sur les réseaux possibilistes ”, thèse de doctorat en informatique, l’Université Paul Sabatier,(2005) .

**[08]** N.ABBAS,’’ Vers une Extension Sémantique d’analyse formelle de concepts : Application à la recherche d’information ’’, Mémoire de magister, (2014).

**[09]** R. EL CHARIF, ‘’ Analyse des paramètres de pondération dans le cadre collections volumineuses ‘’, DEA d’informatique, (2006).

**[10]** M.BEN AOUICHA,’’ Une approche algébrique pour la recherche d’information structurée’’, thèse Doctorat de l’Université Paul Sabatier, (2009).

**[11]** PORTER, ‘An algorithm for suffix stripping Program’’, (1980).

**[12]** S.BOULAKNADEL, “ Traitement automatique des langues et recherche d’information sur une langue arabe dans un domaine de spécialité : Apport des connaissances morphologiques et syntaxique pour l’indexation ”, Université de Nantes, (2008).

**[13]** N. ZIMERLI,’’ Modèle d'accès personnalisé a l'information basse sur les Diagrammes d'Influence intégrant un profil l utilisateur évolutif’’, thèse doctorat l'Université Paul Sabatier de Toulouse II**,(**2009).

**[14]** MILLER,’’ G.A. Miller. Word net: “A lexical data base for English”. In HLT, (1994).

**[15]** H.ALIANE, ‘’ Un système de reformulation de requêtes pour la recherche d’information‘’, Centre de Recherche sur l’Information Scientifique et Technique, Alger, Algérie, (2004).

**[16]** M .CHEIN. ‘’Introduction à la recherche d’information’’, Université Montpellier Master d’informatique, (2005).

**[17]** G .MATHIAS. ‘’Indexation et interrogation de chemins de lecture en contexte pour la Recherche d’Information Structurée sur le Web’’, Thèse pour obtenir le grade de Docteur de l’université Joseph Fourier, (2002).

**[18]** G .SALTON,’’The smart retrieval system: Experiment in automatic document processing’’, (1971).

**[19]** S. E. RROBERTSON and S. WALKER. ‘’On relevance weights with little relevance information’’. In Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval,( 1997).

**[20]** F.BOUBAKEUR. “Contribution à la définition de modèles de recherche d'information
flexibles basés sur les CP-Nets”, thèse de doctorat en informatique, Université Paul Sabatier,( 2008).

**[21]** S.KARBASI, “Modèle de pondération basé sur le rang des termes dans les documents”, thèse de doctorat en informatique, l’Université Paul Sabatier, (2007).

# Liste des figures

# Liste des tableaux

# Liste des abréviations

**RI** Recherche d'Information

**SRI** Système de Recherche d'Information

**B** Bruit

**P** Précision

**R** Rappel

**S** Silence

## Résumé

Notre travail s'inscrit dans le domaine de la Recherche d'Information (RI), qui a pour objectif de fournir à un utilisateur un accès facile à l'information dont il en a besoin. Cette information s'étant située dans une masse de documents textuels, afin d'atteindre cet objectif, nous avons développé une application d'un système de recherche d'information qui doit, stocker, organiser et représenter l'information, et fournir à l'utilisateur les éléments qui correspondants à son besoin en information exprimée par sa requête, comme nous avons intégré quelques langue afin de satisfaire les utilisateurs.

**Mots clés:** Recherche d'information, Systèmes de Recherche d'Information, indexation des documents, pondération des termes, Tf, idf, Tfi-df, Tfc,pertinence, lemmatisation.

**ملخص**

انجازنا يدخل ضمن مجال استرجاع المعلومات و الذي يهدف إلى سهولة وصول المستخدم إلى المعلومات التي تهمه, هذه المعلومات تخزن في كتلة من السجلات النصية,لتحقيق هذا الهدف , قمنا بتطوير نظام استرجاع المعلومات بلغة JAVA الذي من شانه تخزين و تنظيم و تمثيل المعلومات , كما يوفر للمستخدم العناصر التي يبحث عنها حسب طلبه .كما أدرجنا بعض اللغات لتلبية حاجة المستخدم .

**الكلمات المفتاحية** : البحث عن المعلومة ,نظام البحث عن المعلومات , فهرسة المواثيق ,وزن المصطلحات ,Tf, idf, Tfi-df, Tfc, الصلة ,المجموعات الصرفية.

## Abstract

Our work is in the field of Information Retrieval (IR), which aims to provide to the user an easy access to information they are interested in. This one is situated in a mass of textual records. To achieve the goal mentioned, we developed an application based on object-oriented design with the JAVA language of an information retrieval system, that has to store, organize and represent information, so as to make available the elements corresponding to the users' needs on knowledge according to their request. We have also included some languages to satisfy all the users around the world.

**Key words :**Information Retreival,Information Retreival System, indexing document,term weighing,TF,idf,Tf-idf,Tfc, relavance,lemmatisation.