République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DU **G**ENIE **E**LECTRIQUE **E**T D'**I**NFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

Mémoire de Fin d'Etudes
de MASTER ACADEMIQUE
Domaine : **Mathématiques et Informatique**
Filière : **Informatique**
Spécialité : **Ingénierie des Systèmes d'information**

Réalisé par :
BOUBAHA Assia
ALIM Meriem

Thème
Etude comparative des approches de recherche implémentées dans Lucène

Soutenue le / / devant le jury composé de :

Président : M S.SADOU
Examinateur : M H.RADJA
Dirigé par : M Y. YACINE
Proposé par : M^{me} F. AMIROUCHE

Remerciement :

Au terme de la rédaction de ce mémoire, nous estimons qu'il est important d'accorder quelques lignes de reconnaissances à toute personne ayant contribué de prêt ou de loin.

Tout d'abord, nous adressons notre plus profonde gratitude à notre promoteur monsieur YACINE Younes, qui a toujours su orienté nos recherches, et pour tout le temps qu'il nous a accordée.

Nous tenons à remercier également les membres de jury d'avoir accepté d'évaluer notre travail.

Nous tenons à exprimer notre sincère reconnaissance envers Mme AMIROUCHE Fatiha et Mme ACHEMOUKH Farida pour leurs précieuses aides. Nos remerciements vont également à nos amis(es) qui nous ont toujours encouragées que ce soit d'un point de vue moral ou intellectuel.

Dédicaces

A nos chers parents,
A nos frères et sœurs,
A nos familles,
A nos amis(es).

Résumé :

Notre travail s'inscrit dans le domaine de la recherche d'information (RI).Plus précisément, nous nous intéressons aux approches implémentées dans le système de recherches d'informations (SRI) Lucéne qui est l'un des moteurs de recherche les plus connus, les plus utilisés et le plus dynamiques du marché de l'open source. Notre travaille se repose sur l'évaluation de ces approches de recherche sur deux collections de tests AP89 et WSJ, qui a pour but de mettre en évidence les meilleures modèles implémentées dans Lucéne.

Mots clés : recherche d'information, Lucéne, évaluation.

Table des matières :

Table des figures :

Liste des tableaux :

Introduction générale :

La recherche d'information (RI) est une ancienne discipline, qui remonte aux années 1950. Sa problématique peut être vue comme la satisfaction d'un besoin en information d'un utilisateur, qui est exprimé par une requête, sur un ensemble de documents appelé collection. Depuis son apparition, le Web a offert un accès universel aux connaissances et le monde de l'information a été témoin d'une grande révolution numérique. Cette émergence des technologies de l'information et de la communication a produit une masse énorme de documents numériques , ce qui soulève d'importants problèmes pour les utilisateurs notamment pour l'accès aux documents les plus pertinents. Pour faire face à cette révolution, il est devenu primordial de disposer de systèmes de recherche d'information efficaces et performants pour retrouver des informations pertinentes en un temps opportun.

L'emploi des systèmes de recherche d'information (SRI) a permis d'alléger ce problème. Par ailleurs, plusieurs efforts ont été faits pour développer des SRI d'une manière à les rendre plus conviviaux et précis par rapport aux résultats de la recherche d'information. Parmi les meilleurs solutions retenues, Lucéne ; le moteur de recherche qui permet de faire l'indexation et la recherche de contenu.

Depuis le début de son apparition, Lucéne s'est servi des premières approches de la recherche d'information qualifiées de classiques qui se basaient sur une recherche par mot clés où les documents sont représentés comme un ensemble de mots et la pertinence d'un document vis-à-vis d'une requête est souvent estimée en s'appuyant sur les fréquences d'apparition des mots de la requête dans ces mêmes documents. Cependant ces représentations sont très limitées par leurs simples indexations, en outre le traitement de la requête ne prend pas en compte la sémantique et la variation lexicale d'un mot clé. Lucéne s'est alors confronté à un nouveau défi, et a du enrichir sa bibliothèque avec des approches de recherche d'information des plus récentes.

Dans le cadre de notre travail, nous nous focalisons sur les différentes approches de recherche implémentées dans Lucéne afin d'élaborer une étude comparative pour mettre en évidence les modèles de recherche les plus efficaces

Chapitre 1

Généralités sur la recherche d'information

1.1 Introduction :

Depuis son apparition à la fin des années 40, la recherche d'information (en anglais information retrieval) consiste à rendre accessible les connaissances existantes, son but principal est de récupérer des informations à partir d'une collection de documents préalablement stockée pour répondre au besoin informationnel de l'utilisateur.

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. **[1]**

1.2 Définition

Un système de recherche d'information (SRI) est un système qui met en œuvre des outils (programmes informatiques) qui permettent de sélectionner des documents susceptibles de répondre à la requête utilisateur, à partir d'une collection de documents volumineuse.

1.2.1 Concept de base de la recherche d'information

Pour effectuer une recherche, l'utilisateur exprime son besoin informationnel sous forme d'une requête, le SRI compare cette dernière à sa collection qui peut contenir des millions de documents, plus la requête et le document ont de mots en commun, plus le document est considéré comme étant pertinent. Ensuite il sélectionne dans un temps raisonnable une dizaine voir des centaines ou des milliers de documents pertinents pour répondre à la requête utilisateur.

Dans ce qui suit, nous allons définir les éléments principaux de la recherche d'information que nous venons d'évoquer.

1.2.2 Le document :

Un document est toute unité qui peut constituer une réponse à une requête d'utilisateur, il est accessible et exploitable par le SRI, il peut être représenté sous plusieurs formats :

- Non structuré : textes, images, vidéos, sons ;
- Semi structuré : HTML, XML, JASON ;
- Structuré : base de données.

1.2.3 La collection de document :

C'est un ensemble de documents stockés et organisés qui contient des informations exploitables et accessibles par l'utilisateur, sa gestion est assurée par le SRI.

1.2.4 La requête :

La requête exprime le besoin en informations de l'utilisateur, elle est écrite dans un langage d'interrogation compréhensible par le SRI, on site parmi ces langages :

- Le langage booléen : la requête est un ensemble de mots clés séparés par des opérateurs logiques (non, ou, et) ;
- Le langage naturel : la requête est exprimée avec un ensemble de mots clés dans un langage libre.

1.2.5 La pertinence :

La notion de pertinence est très importante dans la RI car toutes les évaluations s'articulent autour d'elle, elle est considérée comme une correspondance (ou une similarité) entre un document et une requête, par conséquent, dans un document pertinent, l'utilisateur doit pouvoir trouver les informations dont il a besoin, mais en pratique il est difficile de savoir son jugement, c'est pour cette raison, qu'on a introduit les deux notions suivantes :

- La pertinence système : Effectuée par le SRI qui évalue la similarité entre le document et la requête, cette pertinence est objective et déterministe, elle ne tient pas en compte le jugement de l'utilisateur.

- La pertinence utilisateur : Elle dépend du jugement utilisateur sur les documents retournés en réponse par le SRI, elle est dite subjective car un même document retourné en réponse à une même requête, peut être jugé différemment par deux utilisateurs différents.

1.2.6 Le modèle de représentation

Avant d'être exploités par le SRI, la requête utilisateur et les documents de collection sont d'abord traduit vers une forme plus structurée appelée « représentation interne », ensuite le SRI fait une comparaison entre ces deux représentations.

1.3 Processus de la recherche d'information :

Le processus de recherche d'information est mis en œuvre par un Système de Recherche d'information (SRI) afin de répondre à une requête utilisateur qui se décompose, suivant le modèle de recherche implémenté par le SRI, en deux ou trois processus : le processus d'indexation, le processus d'appariement et éventuellement le processus de reformulation de la requête. (figure.1)

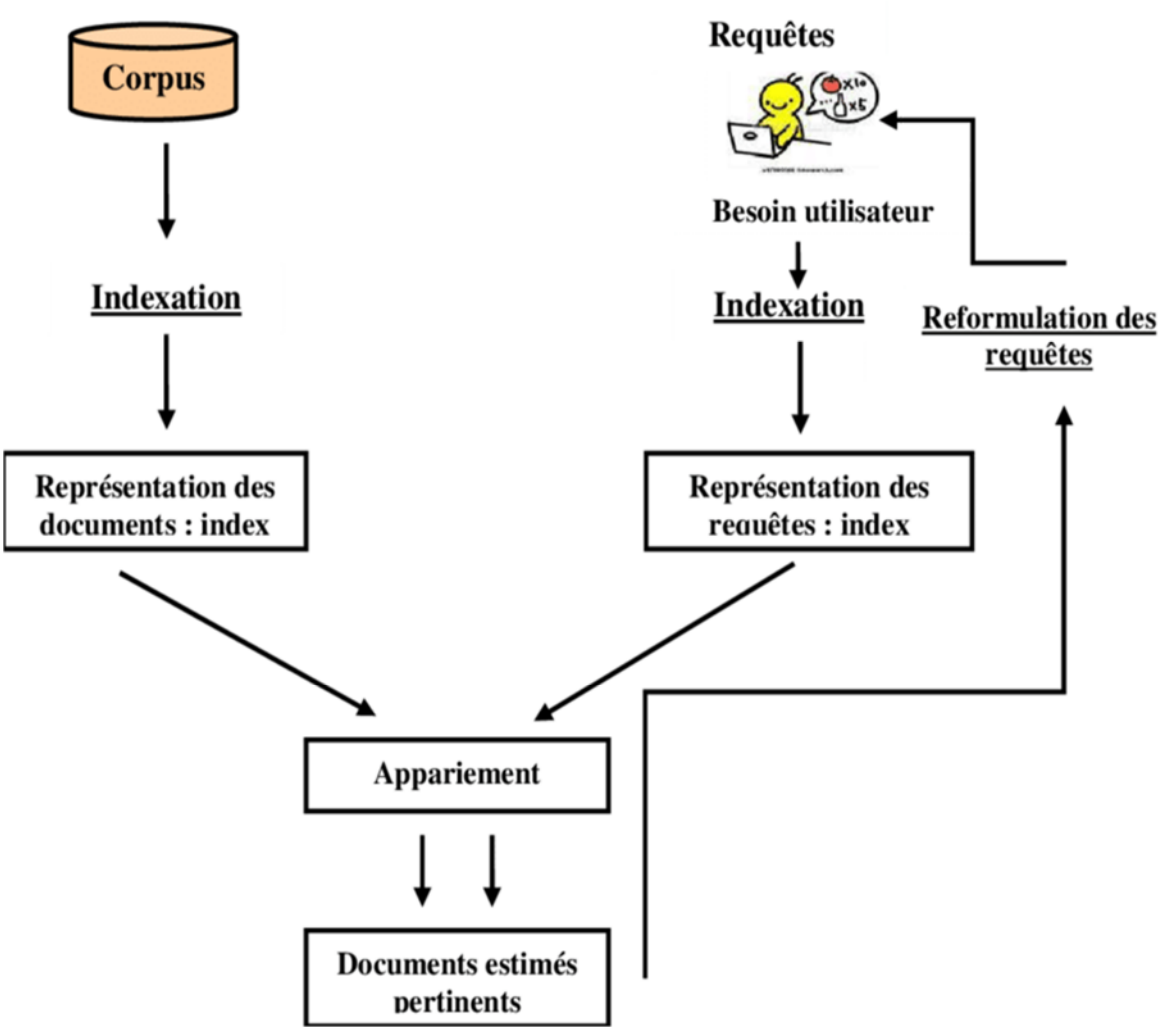

Figure 1.3.1: Processus de recherche d'information

Dans ce qui suit, nous présentons chacun de ces processus.

1.4 Le processus d'indexation :

Cette étape consiste à identifier pour chaque document les termes importants, puis à exploiter ces termes comme index pour accéder rapidement aux documents. L'un des objectifs de l'indexation est donc de permettre de retrouver rapidement les documents contenant les termes (mots-clés) de la requête.

L'indexation peut être manuelle, automatique ou semi-automatique :

- **Indexation manuelle :** chaque document est analysé par un expert du domaine, qui identifie les mots clés appelés descripteurs.
- **Indexation automatique :** fait appel aux robots d'indexation, ce qui rend le processus d'indexation complètement automatisé.
- **Indexation semi-automatique** : chaque document est d'abord analysé par un processus automatique qui construit son descripteur initial, puis le spécialiste du domaine intervient pour le choix final des termes à partir du descripteur initial.

Dans la RI moderne, l'indexation automatique est l'approche la plus utilisée, nous la décrivons dans ce qui suit.

1.5 L'indexation automatique :

C'est la technique d'indexation la plus utilisée, elle est la plus adaptée pour les bases documentaires volumineuses grâce à son processus entièrement automatique avec une capacité de traitement plus rapide que les autres approches. Le SRI choisit d'indexer les termes les plus représentatifs dans chaque document. Ce traitement est réalisé en plusieurs étapes: l'analyse lexicale, l'élimination des mots vides, la normalisation des termes et la pondération.

1.5.1 L'analyse lexicale (tockenization) :

La première étape de l'indexation automatique consiste à identifier et extraire du document les unités lexicales (des termes). Une unité lexicale est définie comme étant une suite de caractères délimitée par des séparateurs (blanc, signe de ponctuation, caractères spéciaux…), chaque unité lexicale définit un mot ou un groupe de mots.

1.5.2 L'élimination des mots vides :

Elle consiste à éliminer les mots non porteurs d'information pour la recherche et les termes très fréquents dans les documents de la collection, tel que les prénoms personnels, les conjonctions, les prépositions et les articles. Les deux techniques principalement utilisées pour l'élimination des mots vides sont :

- L'utilisation d'une liste prédéfinie de mots vides, aussi appelée anti-dictionnaire ou stoplist ;
- L'utilisation de mesures statistiques sur la collection de documents pour déterminer les mots les plus fréquents ou les mots rares, qui sont considérés des mots vides.

Cette étape permet de réduire la taille de l'index, et donc gagner de l'espace mémoire et améliorer la fiabilité du SRI en termes de qualité logiciel (temps d'exécution) et performance.

1.5.3 La normalisation des termes :

L'idée qui conduit à utiliser la normalisation est de pouvoir indexer un ensemble de mots par un seul mot qui représente le même concept, ce traitement est réalisé grâce aux procédures de lemmatisation et de troncature.

La lemmatisation consiste à regrouper les variantes d'un mot (mots de la même famille) et extraire la forme canonique (le lemme) de chacun des termes puis remplacer chaque mot par son lemme. Pour ce faire, on considère la forme à l'infinitif des verbes (marchais, marchera, marchâmes → marcher), et la forme au masculin singulier pour les noms et les adjectifs (petits, petite, petites → petit).Cette méthode demande des traitements linguistiques particuliers pour prendre en charge certaines exceptions telles que les variations des verbes de troisième groupe.

La troncature consiste à éliminer les suffixes (et rarement les préfixes) des mots pour obtenir leurs racines (information, informatique, informaticiens → informat). Contrairement à la lemmatisation, la troncature ne dépend pas la langue, en plus son processus est plus rapide et l'index obtenu est plus petit. En effet, il existe d'autre méthode de normalisation tel que

« l'algorithme de Porter » conçu pour la langue anglaise.

1.5.4 La pondération :

La pondération est l'une des fonctions fondamentales en RI, elle permet d'attribuer à chaque terme t_i un poids qui représente son importance dans un document 'd', le poids augmente proportionnellement au nombre d'occurrence du terme dans le document. On distingue deux types de pondération :

- **La fréquence locale (tf)** (terme frequency) : elle calcule la fréquence d'apparition d'un terme dans un document, plus un terme est important (fréquent), plus sa fréquence est grande. La fonction de pondération la plus utilisée est tf_{ij} qui représente le nombre d'occurrence un terme t_i dans un document d_j**.**

- **La fréquence globale (idf)** (inverse document frequency) : Cette fréquence calcule l'importance du terme dans tous le corpus, le poids le plus important est attribuée au terme qui apparait dans peu de document de la collection, ici il s'agit de valoriser les termes rares et les considérer comme représentatifs du contenu des documents. Le facteur de pondération globale est calculé selon :

$$idf(w) = \log\left(\frac{N}{df(w)}\right)$$

Tel que : df(w) : le nombre de documents contenant le terme 'w' ;

N : le nombre total de document dans la collection.

En général les méthodes de pondération sont construites par la combinaison de pondération globale et de pondération locale pour obtenir à la fois un terme considéré important dans une collection (avec IDF élevé) et qui est fréquent dans l'un de ses documents (Tf élevé). La méthode la plus utilisée est la méthode de TF*IDF exprimé comme suit :

$$W_{ij} = tf_{ij} \times idf_j$$

Avec : tf_{ij} la fréquence d'occurrences du terme t_j dans le document d_i.

idf_j la fréquence documentaire inverse du terme t_j.(des documents contenant t_j)

1.6 Le processus d'appariement requête-document :

La comparaison entre le document et la requête revient à calculer un score représentatif de la ressemblance entre le document et la requête. Ce score de similarité entre le document et la requête est donné par une fonction nommée Retrieval Status Value. Elle est notée RSV(d,q), où 'd' est un document et 'q' est une requête. Traditionnellement le système de recherche retourne à l'utilisateur une liste de documents classés par RSV. Cette fonction est fondamentale pour la RI car c'est elle qui détermine comment comparer la requête aux documents indexés. Le processus d'indexation et la fonction d'appariement constituent les deux éléments essentiels du modèle de recherche. Avant de présenter les différents modèles de la RI, il est nécessaire d'introduire un dernier concept celui de reformulation de requête.

1.7 Le processus de la reformulation de la requête :

En plus des deux processus de base décrits précédemment, un SRI peut intégrer un processus de reformulation de la requête. Ce processus a pour but d'améliorer la performance du système en offrant un mécanisme de raffinement de la requête utilisateur en se basant sur le corpus documentaire ou sur les résultats d'une première recherche initiée par la requête initiale.

Il existe deux approches principales pour la reformulation de requête :

— **L'expansion automatique des requêtes :** consiste à construire une nouvelle requête Q_1, qui se rapproche du besoin informationnel de l'utilisateur, en étendant la requête initiale Q_0 avec des mots du corpus qui leur sont sémantiquement liés.

— **La réinjection de pertinence (Relevance Feedback) :** consiste à construire une nouvelle requête Q_1, en étendant la requête initiale Q_0 à partir des résultats d'une première recherche initiée par Q_0.

1.8 Évaluation des systèmes de recherche d'information :

Depuis la naissance du domaine de la RI, l'évaluation des modèles et méthodes proposés a toujours été un centre d'intérêt important. Pour cela des mesures de qualité des systèmes de recherche ainsi que des ensembles de données ont été développés afin de tester ces systèmes sur une base commune. La qualité d'un système doit être mesurée en comparant les réponses

du système avec les réponses que l'utilisateur espère. Plus les réponses du système correspondent à celles que l'utilisateur espère, meilleur est le système.

1.8.1 Mesure d'évaluation :

Les mesures d'évaluation permettent d'estimer l'efficacité d'un SRI. Le but étant de mesurer, pour chaque requête la capacité du système à retourner des documents pertinents. Dans ce qui suit nous définissons deux mesures d'évaluations : le rappel et la précision, la figure ci-dessous représente le processus de recherche d'information. [2]

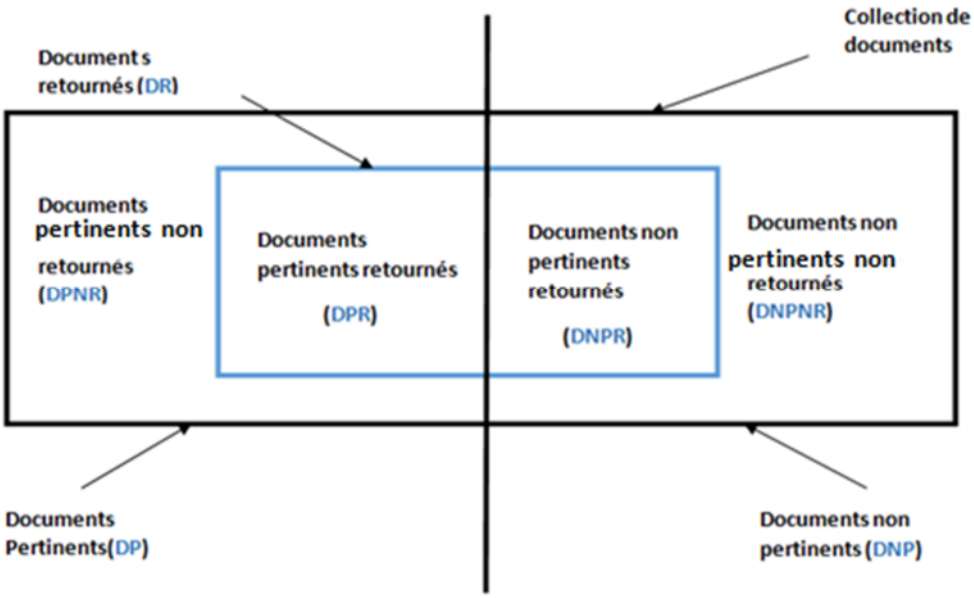

Figure 1.8.1 : Répartition des documents d'une collection suite à une requête.

- **Le rappel :** représente le ratio de documents pertinents retournés par rapport à l'ensemble des documents pertinents de la collection documentaire [3]

Où :

$$R = \frac{\text{nombres de documents pertinents retournés (DPR)}}{\text{nombres de documents pertinents total(DP)}}$$

Le rappel permet aussi de définir le **silence documentaire** qui représente la proportion des documents pertinents non retrouvés par le système qui se calcule comme suit :

$$\text{Silence} = 1 - \text{rappel}$$

- **La précision :** représente la proportion de documents pertinents parmi les documents sélectionnés.

Où

$$P = \frac{\text{nombre de documents pertinents retournés(DPR)}}{\text{nombre total de documents retournés par le systéme(DR)}}$$

La précision permet aussi de définir le **bruit documentaire** qui mesure la proportion de documents non pertinents retournés par le système qui se calcule comme suit :

Bruit = 1- précision

On dit qu'un SRI est idéal lorsqu'il retourne tous les documents pertinents (rappel=1) et tous les documents pertinents total (précision=1), mais en pratique ce n'est pas le cas parce qu'il y a une relation inverse entre ces deux quantités, lorsque l'une augmente l'autre diminue. En outre, pour faire le rappel il suffit de sélectionner tout le corpus, mais ainsi la précision sera très faible comme illustré dans la figure :

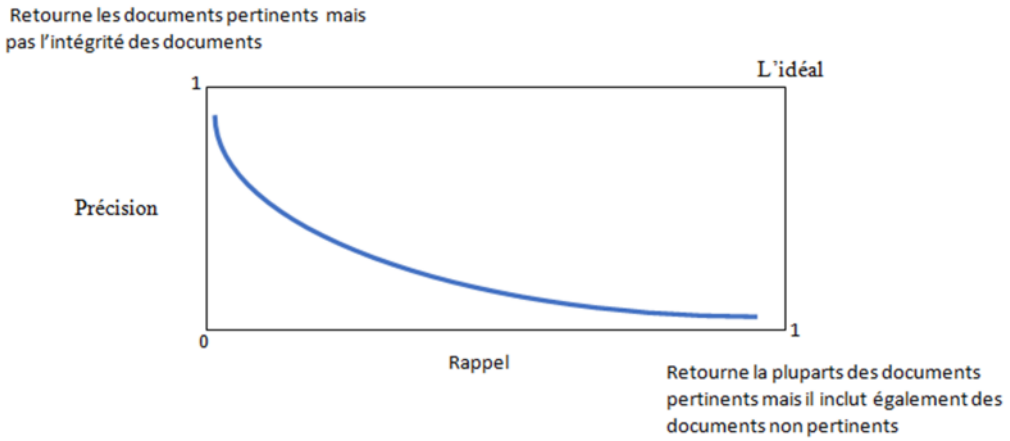

Figure 1.8.2: Rappel et précision dans le corpus [4]

Par ailleurs, l'un des moyens possible pour combiner le rappel et la précision en une seule valeur est la courbe reliant entre le rappel et la précision appelée 'Courbe Rappel / Précision '

1.8.2 La courbe rappel-précision :

La plupart des approches d'évaluation des SRI se basent sur les deux mesures de rappel et de précision. En effet, plus ces deux valeurs sont proches de un (1) pour un système donné, plus ce système est performant. Or, elle existe une forte corrélation entre ces deux valeurs. Et, en pratique, plus l'une des deux valeurs augmente plus l'autre diminue. Ainsi en faisant varier l'une des valeurs on peut obtenir une courbe représentant la précision en fonction du rappel (ou inversement).

Il est possible, grâce à cette représentation, de comparer deux systèmes de recherche d'information. Particulièrement, si les deux courbes ne se croisent pas, on peut déduire que le système dont la courbe se trouve au-dessus de l'autre, est plus performant (car il offre un taux de rappel et de précision plus élevé).
En outre, plus la courbe rappel-précision d'un système décroit tardivement, plus ce système est performant.

Un exemple d'une courbe rappel-précision est présenté dans la figure 2

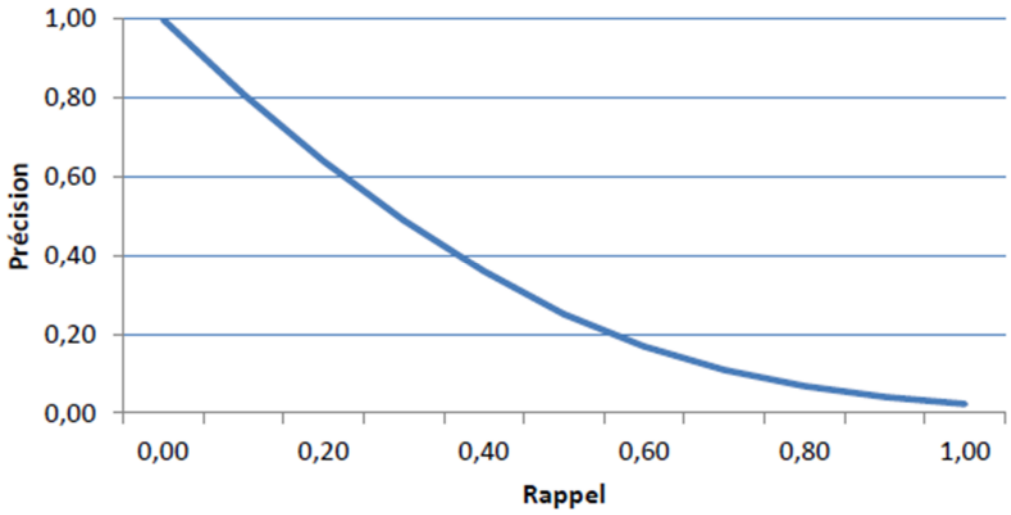

Figure 1.8.3: Courbe Rappel / Précision

1.8.3 Mesures d'évaluation ordonnancées:

Cependant, que ça soit une recherche effectuée par un utilisateur ou une évaluation des performances des approches de la RI, le système doit être en mesure de retourner les documents pertinents avant les documents non pertinents, d'où vient la notion d'ordonnancement. Pour ce faire, l'utilisation des métriques adéquates qui tiennent en compte le rang du document est nécessaire. Nous présentons dans ce qui suit les mesures principales utilisées dans ce contexte.

1.8.3.1 Précision à K :

Soit 'r' le nombre de documents pertinents retournés par le système au rang K(les K premiers documents retournés), La précision corresponds à :

$$P@\ K = \frac{r}{k}$$

Par exemple, P @10 correspond au nombre de résultats pertinents parmi les 10 premiers documents retournés par le système.

1.8.3.2 Courbe interpolée Rappel/précision :

En général, la courbe rappel/précision est basée sur 11 points standard du rappel, de 0 % jusqu'à 100 % par pas de 10 %. En pratique ces valeurs du rappel peuvent ne pas être atteintes exactement ; les valeurs de la précision doivent donc être interpolées. La règle d'interpolation est la suivante : la valeur interpolée de la précision pour un niveau de rappel 'i' est la précision maximale obtenue pour un rappel supérieur ou égal à i :

$$P(r_j) = max_{r_j+1>r\geq r_j} P(r)$$

Avec :

r_j est la référence au j^{ieme} niveau standard de rappel ;

$P(r_j)$ est la précision interpolée au niveau standard de rappel r_j.

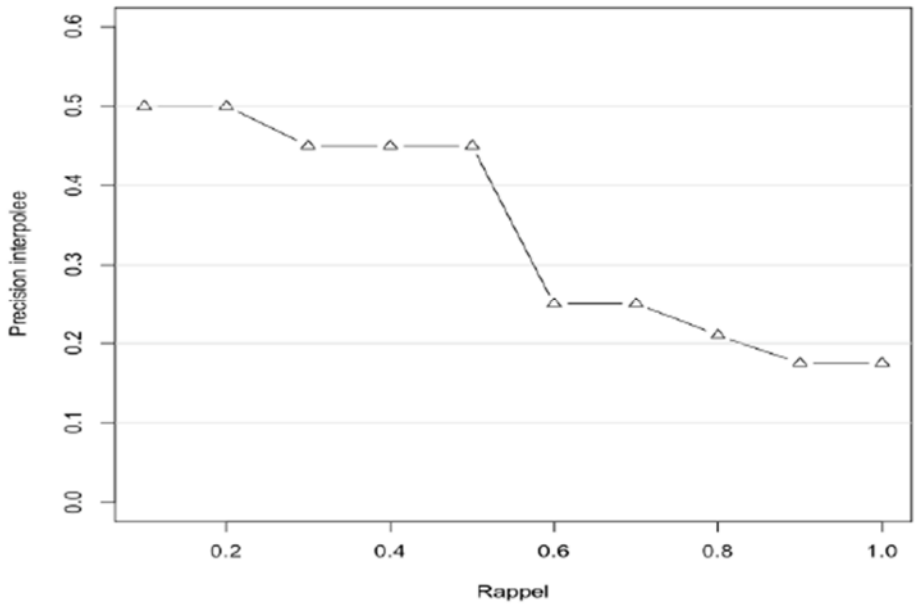

Figure 1.8.4: Exemple de la courbe Rappel / Précision interpolée

Les courbes rappel/précision interpolées sont également utilisées pour comparer les performances de différents systèmes ou algorithmes de recherche.

1.8.3.3 La précision moyenne :

La précision moyenne est la moyenne arithmétique des valeurs de précision P pour un système de recherche d'informations sur un ensemble de 'Q' sujets de requête **[5]**.Elle favorise les systèmes qui retournent les documents pertinents rapidement, elle peut être exprimée comme suit:

$$\text{AV\_P(q)} = \frac{1}{Q} * \sum_{q=1}^{Q} P$$

1.8.3.4 La R_précision :

Pour une requête donnée 'q', le système génère une valeur de classement unique en calculant la précision au rang R, où R est le nombre de documents pertinents pour la requête donnée. Cependant, si le système retourne 'r ' documents pertinents parmi les R premiers documents retournés, alors R -Précision est donnée comme suit :

$$\text{R\_prec} = \text{P@ R} = \frac{r}{R}$$

1.8.3.5 MAP (Mean Average Precision) :

La moyenne des précisions moyenne est la moyenne arithmétique des valeurs de la précision moyenne pour un système de recherche d'informations sur un ensemble de n requêtes à chaque fois qu'un document est retrouvé. Sa formule est la suivante :

$$\text{MAP} = \sum_{q=1}^{Q} \frac{\text{AV\_P(q)}}{Q}$$

Avec AV_P(q) est la précision moyenne pour une requête donnée ;

Q le nombre de requêtes dans l'ensemble.

1.8.3.6 GMAP (Mean Average Precision) :

Introduite pour la première fois dans la conférence TREC 2004, GMAP est la moyenne géométrique des valeurs de précision moyennes sur un ensemble de 'n' requêtes données **[6]**. La moyenne géométrique est utilisée comme alternatif à la moyenne arithmétique car elle gère la complexité des requêtes de manière plus cohérente **[7]**.La GMAP est définie comme la racine nième du produit de n valeurs :

$$\text{GMAP} = \sqrt[n]{\prod_n MAP}$$

1.8.3.7 MRR (Mean Reciprocal Rank):

Mean Reciprocal Rank est la moyenne des rangs réciproques des résultats pertinents retournées pour un échantillon de requêtes Q, sachant que le rang réciproque d'une réponse à la requête est l'inverse multiplicatif [1]du rang de la première réponse correcte: 1 pour la première position, 1/2 pour la deuxième position,1/3 pour la troisième et ainsi de suite.

La MRR se calcule de la manière suivante :

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{Rank_i}$$

Avec : $|Q|$ le nombre de requêtes ;

Rank_i : Fait référence à la position de classement du premier document pertinent pour la $i^{ème}$ requête.

[1] Inverse multiplicatif pour un nombre x est 1/x

1.8.3.8 La mesure BPref (Binary Preference):

La mesure BPref est conçue pour les situations où les jugements de pertinence sont loin d'être complets. Elle a été introduite dans la piste TREC 2005, BPref calcule une relation de préférence pour savoir si les documents jugés pertinents sont récupérés avant les documents jugés non pertinents. Ainsi, Contrairement aux autres mesures, BPref se focalise sur les documents réellement jugés [8]. La mesure Bpref est définie comme suit :

$$BPref = \frac{1}{R}\sum\left(1 - \frac{n\ ranked\ higher\ than\ r}{min(R,N)}\right)$$

Où :

R : est le nombre de documents jugés pertinents ;

N : est le nombre de documents jugés non pertinents ;

r : est un document pertinent récupéré ;

n : est un membre des premiers R documents non pertinents récupérés ;

" n ranked higher than r " signifie " les n classés avant r " c'est-à-dire : les documents non pertinents classés avant le document jugé pertinents.

Lors de la comparaison des systèmes sur un test de collections avec jugements complets, MAP et BPref sont considérés comme équivalents. Mais lorsque les jugements sont incomplets, BPref est affiché être plus stable.

1.9 Compagnie d'évaluation :

Plusieurs compagnies ont été créées dans ce contexte pour définir les critères utilisés dans l'évaluation, tel que Cranfield et TREC qui constituent le modèle dominant que nous décrivons par la suite.

1.9.1 Les compagnies TREC :

Le Texte REtrieval Conference **TREC**[2] est une série d'ateliers portant sur de différentes recherches d'information. Fondée en 1992 par l'Institut national des normes et de la technologie **NIST**[3] (National Institue of Standards and Technology), son objectif est d'encourager le travail au sein de la communauté de la recherche d'information, en fournissant des méthodologies d'évaluation des SRI ainsi que l'infrastructure nécessaire pour l'évaluation tel que les Framework d'évaluation et les collections de tests. Ces dernières sont composées d'un ensemble de documents ainsi qu'un ensemble de requêtes et de jugements de pertinence.

1.10 Conclusion :

Ce chapitre a été consacré à la présentation des concepts fondamentaux de la recherche d'information. Dans un premier temps, nous avons développé les étapes du processus de la RI ; notamment le processus de l'indexation automatique. Aprés quoi nous avons vu les principaux critères d'évaluation d'un SRI.

Le chapitre suivant portera une étude approfondie sur le moteur de recherche Lucène, l'une des plus grandes bibliothèques d'indexation open source.

[2] https://trec.nist.gov
[3] *www.nist.gov*

Chapitre 2

Présentation du moteur de recherche Lucène

2.1 Introduction :

Porté par la fondation Apache en mars 2000 **[9],** Lucène est la solution retenue pour de nombreux exploitants de sites Web pour l'indexation et la recherche d'information. Ce projet, que Doug Cutting a débuté comme un passe-temps à la fin des années 1990 **[10],** s'est depuis transformé à l'outil de recherche le plus connu, le plus utilisé et le plus dynamique du marché de l'open source. De nos jours de nombreuses entreprises ont intégré Apache Lucène, en ligne ou hors ligne, entre autre, les recherches sur Twitter sont entièrement basées sur Lucène et Wikipédia l'avait implémenté comme fonction de recherche pendant quelques années.

Lucène peut se définir comme une bibliothèque de recherche et d'indexation de contenus (Information Retrieval Library) qui se base sur le concept d'indexation automatique, c'est aussi le cas de Google et d'autres moteurs de recherche, mais ces derniers se limitent à l'information provenant du World Wide Web, contrairement à Lucène qui peut être utilisé dans n'importe quel scénario et configuré en fonction de vos besoins précis.

2.2 Définition du Lucène :

Lucène est une bibliothèque de programmes publiée par l'Apache Software Fondation. C'est un logiciel gratuit et libre. À l'origine, Lucène a été écrit entièrement en Java par DOUG Cutting. Cependant il existe maintenant des ports vers d'autres langages de programmation **[10]**.

Lucène est une recherche plein texte. Cela signifie tout simplement qu'un programme recherche un ou plusieurs termes définis par l'utilisateur dans une série de documents texte. Cela montre que Lucène n'est pas seulement utilisé dans le contexte du World Wide Web, même si les fonctions de recherche sont omniprésentes sur le Web.

Lucène peut également être utilisé pour les archives et les bibliothèques. Ce moteur ne recherche pas seulement des documents HTML, mais travaille aussi avec des emails ou même des fichiers PDF.

2.3 Fonctionnalités de Lucène :

Lucène propose des fonctionnalités puissantes via une API simple **[11]:**

- **Indexation évolutive et hautes performances :**
  - Plus de 150 Go / heure sur un matériel moderne ;
  - Petite exigence de RAM (seulement 1 Mo) ;
  - Indexation incrémentielle [1] aussi rapide que l'indexation par lots[2] ;
  - Taille de l'index environ 20 à 30% de la taille du texte indexé.

- **Algorithmes de recherche puissants, précis et efficaces :**
  - les meilleurs résultats sont retournés en premier.

- **Disponibilité :**
  - Disponible en tant que logiciel open source sous la licence Apache qui permet d'utiliser Lucène dans des programmes commerciaux et open source ;
  - Lucène est extensible, tout en conservant une architecture simple et cohérente ;
  - Implémentations dans d'autres langages de programmation disponibles et compatibles avec les index.

[1] *L'index est mis à jour.*
[2] *Indexation par tableau.*

2.4 Architecture et organisation de Lucène :

L'architecture de Lucène est illustrée dans la Figure 2.

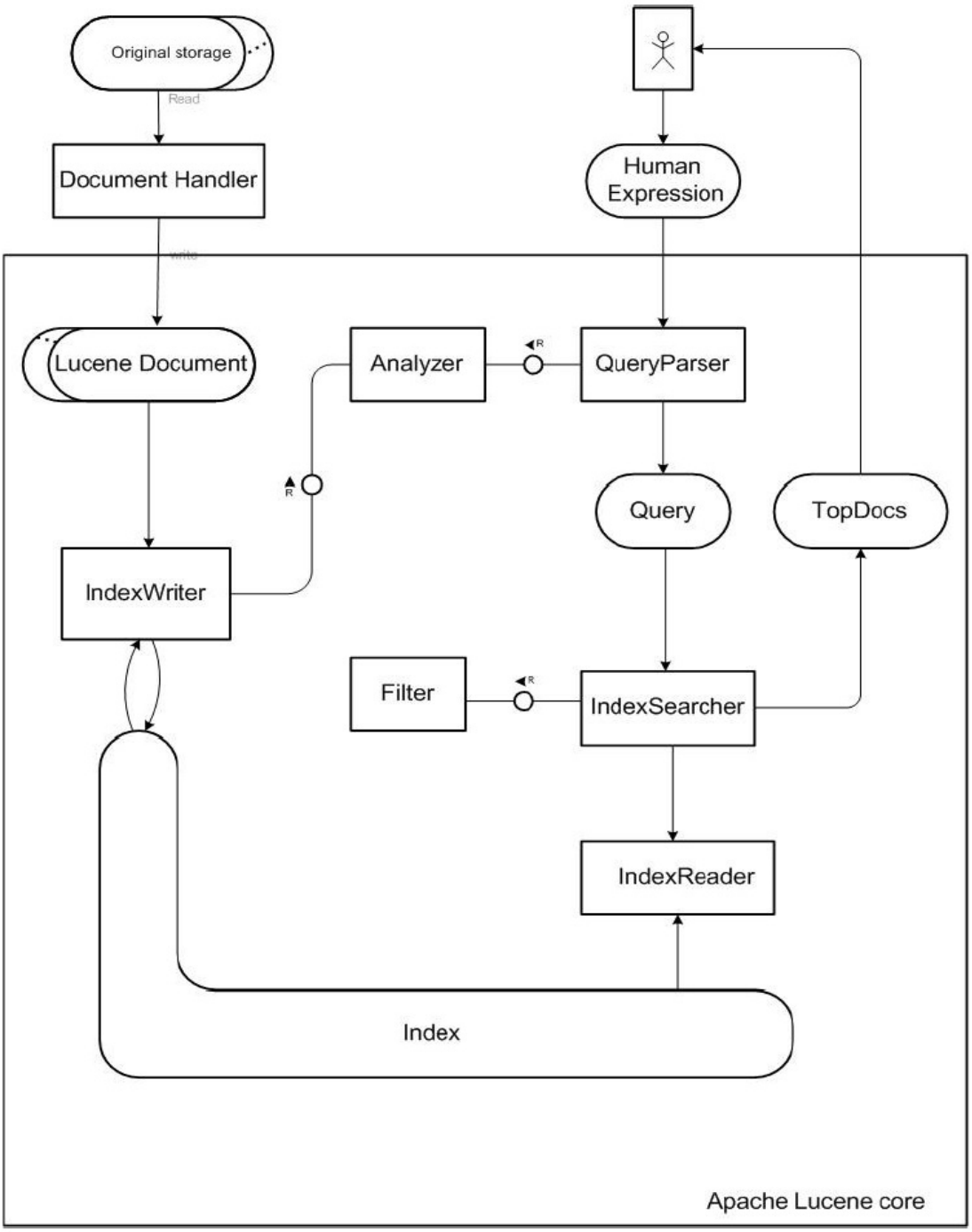

Figure 2.4.1: Architecture de Lucène

Toute Application utilisant Apache Lucène doit tout d'abord transformer ses données d'origine, en documents Lucène. Pour cela, une interface de gestionnaire de documents appelée « Document Handler » est nécessaire, celle-ci est fournie par la bibliothèque de contribution Lucène. Document Handler permet l'extraction d'informations telles que le contenu textuel, les nombres et les métadonnées à partir de documents originaux et de les fournir en tant que documents Lucène. Ceux-ci sont utilisés pour un traitement ultérieur lors de l'indexation et de la recherche.

Chaque type de document commun comme HTML, PDF, XML, etc a besoin d'un analyseur de document spécifique pour extraire son contenu. Ils sont disponibles gratuitement sur le Web. Certains d'entre eux sont :

- JTidy: un analyseur HTML ;
- PDF box: un analyseur de documents PDF ;
- SAX: un analyseur XML.

Une fois qu'un document Lucène est créé, IndexWriter est le composant suivant chargé d'analyser et de stocker les documents Lucène dans l'index. Cela se fait selon des attributs particuliers. L'indexWriter utilise un ou plusieurs Analyzer comme stratégie d'écriture d'index.

L'analyseur purge les documents Lucène des contenus inutiles tels que l'espace, le trait d'union, les mots vides et bien plus encore en fonction du ou des analyseurs choisis. À la fin du processus d'analyse, un document Lucène est divisé en termes utilisés pour la recherche.

Pour effectuer une recherche dans l'index, l'utilisateur doit fournir une expression lisible par l'homme appelée chaîne de requête. Ensuite elle sera transformée par le QueryParser en un objet de type Query qui doit être analysé, puis assigné à IndexSearcher qui est au cœur du processus de recherche. Il utilise IndexReader pour accéder à l'index et pour récupérer tous les termes qui correspondent aux termes de l'objet Query, et renvoie les topdocs (les documents jugés pertinents) comme résultat de la recherche. Un filtre peut être utilisé pour autoriser ou interdire un ou plusieurs termes dans les résultats de recherche.

2.5 Packages de Lucène :

Comme la montre la figure ci-dessus [12], Lucène se découpe en 7 packages :

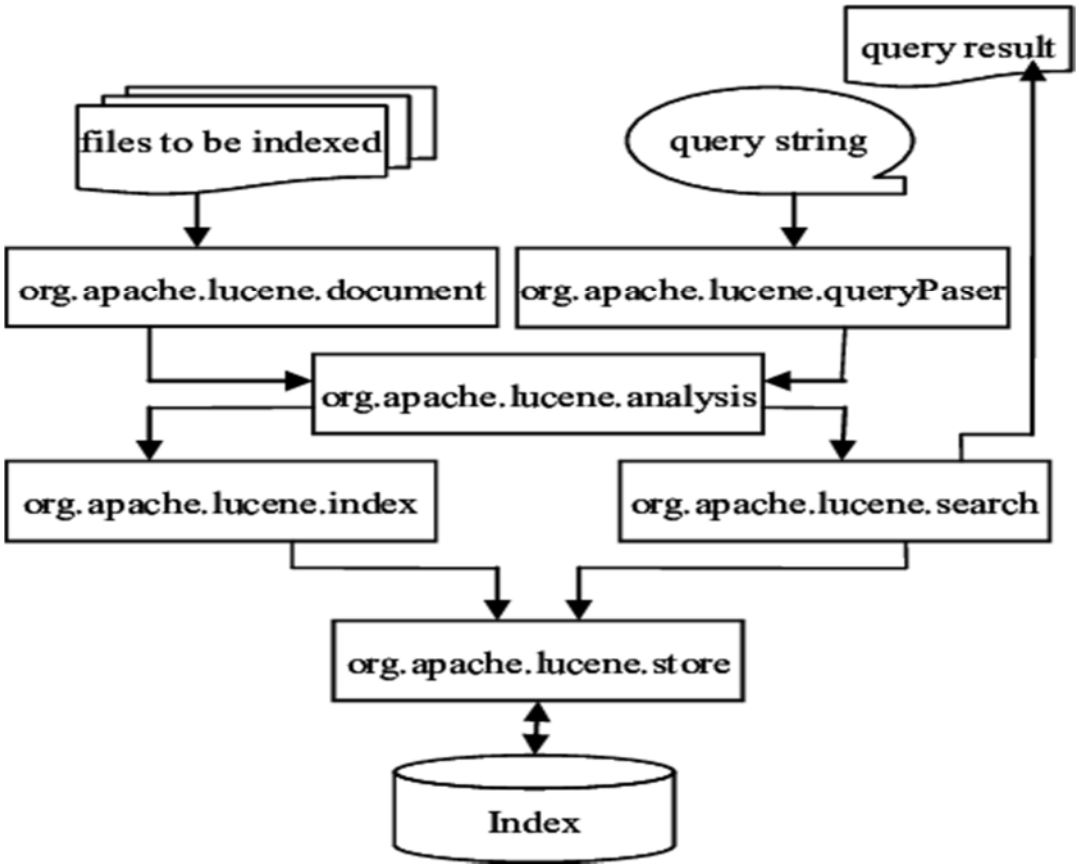

Figure 2.5.1: Les packages de Lucène.

- **Org.apache.Lucene.document**: Contient des classes relatives aux documents, tel que la classe : Document. Cette classe représente un rassemblement de champs(Fields), ainsi les métadonnées sont indexées et stockées séparément comme des champs d'un document.

- O**rg.apache.Lucene.analysis** : Ce package fournit des classes qui peuvent convertir du texte en un élément indexable, à l'exclusion d'autres informations non intéressantes. Il propose aussi une multitude de classes qu'on peut combiner. Ce qui permet de créer selon le type de documents qu'on a, des analyseurs personnalisés afin de générer nos tokens à partir des flux de caractères.

- **org.apache.Lucene.index** : Contient les classes qui génèrent les indexes, il fournit en fait tous les éléments pour les manipuler. Il contient une multitude de "writer" et de "reader" spécialisée sur la lecture et l'écriture des différents composants du format d'indexation de Lucène.

- **org.apache.Lucene.queryPaser** : Son rôle est d'analyser les requêtes afin de les générer sous forme d'objets Query qui pourront ensuite être réutilisés par le parseur.

- **org.apache.Lucene.search** : Ce package se charge de fournir les objets à chercher dans les indexes. Il fournit les classes suivantes :
  - IndexSearcher : cette classe se charge de l'ouverture de l'index en lecture seulement.
  - Query : c'est la méthode la plus basique d'interrogation de Lucène, elle est utilisée pour égaliser les documents qui contiennent des champs avec des valeurs spécifiques.
  - Hits : la classe Hits est un conteneur d'index pour classer les résultats de recherche de document.

Lorsqu'un utilisateur utilise un objet Searcher (et plus précisément IndexSearcher), il utilise la méthode search (Query, filtre) qui retourne un objet de type Hits. Une classe Hits est un cache de résultats, autrement dit, plutôt que de tout charger en mémoire, les résultats qui ne seront pas forcément utilisés lors d'une recherche seront dissipés. L'objet Hits va contenir en lui le parcourt des résultats, la requête et le filtre.

- **org.apache.Lucene.store** : Pour s'abstraire de la source de donnée, Lucène s'utilise derrière l'interface Directory. Ce package représente donc une couche d'abstraction d'entrée sortie.

On y trouve les classes suivantes :

— Directory : Les fichiers peuvent être écrits une fois, lorsqu'ils sont créés ils ne peuvent être ouverts qu'en lecture ;

— FSDirectory : c'est une classe qui étend de la classe Directory, elle sert à stocker des fichiers d'index dans le système de fichiers.

- **org.apache.Lucene.utils** : contient des classes qui peuvent être utiles aux développeurs utilisant ou développant l'api, et qui sont utilisées dans les autres packages.

2.6 Fonctionnement de Lucène :

Lucène se comporte comme une couche intermédiaire entre les données à indexer et le programme utilisateur. Pour ce faire, il indexera des objets appelés documents, ces derniers pourrait être un texte Word, un fichier PDF, un ensemble de fichiers, une page web sur un serveur distant, des informations stockées dans une base de données …etc. Et à partir des index, il permettra une recherche rapide et efficace dans ces documents, avec une seule exigence : le document original doit pouvoir être converti en fichier texte.

Pour ce moteur de recherche, l'index est le facteur décisif pour la recherche, il est tellement important qu'il est considéré comme le cœur de Lucène : tous les termes de tous les documents sont stockés dedans et toutes les requêtes de recherche s'effectuent sur l'index. Un tel « Inverted Index **»** n'est en principe qu'un tableau dans lequel, pour chaque terme, la position correspondante est enregistrée, c'est une compilation de mots-clés et de propriétés identifiant des documents. Pour construire un index, il faut d'abord procéder à une extraction, cela signifie parcourir tous les documents pour extraire les termes et les sauvegarder dans l'index. Lucène permet aux utilisateurs de configurer cette extraction individuellement ; Lors de la configuration, les développeurs auront la possibilité de choisir les champs à inclure dans l'index **[10].**

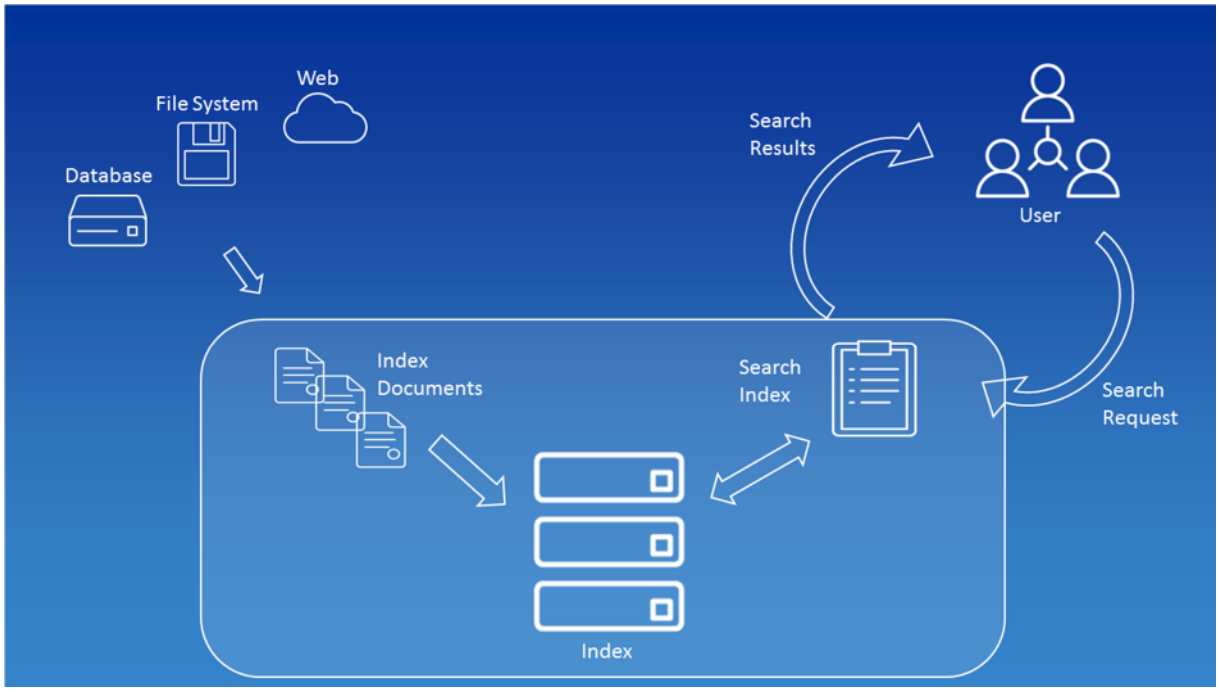

Figure 2.6.1: Le cœur du fonctionnement de Lucène .

Lucène travaille sur les documents sous toutes leurs formes, Cependant, les documents eux-mêmes contiennent, selon l'approche de Lucène, des champs qui peuvent contenir des attributs relatifs aux documents tel que le nom de fichier, le titre du document ou le nom de l'auteur, en effet chaque champ possède un nom unique et une valeur. Dans un autre cas, nous pourrons indexer une page HTML grâce aux mots-clés contenus dans une balise <meta> et conserver en tant que propriétés son URL complète et son titre.

Lors de la création d'un index, la bibliothèque crée un nouveau répertoire contenant plusieurs fichiers appelés des segments. Les classes nécessaires à la création d'un nouvel index, ou à l'extension d'un existant, se trouvent dans les packages "org.apache.Lucene.index" et "org.apache.Lucène.analysis.standard". Le premier contient l'outil de création de l'index tandis que le second accueille les analyseurs de texte. En effet, pendant la création de l'index, on commence par la « tokenisation » qu'on a défini dans le chapitre précédant. Dans cette étape, on s'éloigne du niveau des bits et on s'intéresse au contenu lisible par l'homme. Pour une machine, un document est avant tout un assemblage de données constitué d'une chaîne de caractères : lettres, ponctuation et espaces. Dans cette étape, Lucène fait appel à des tokenizers et des filtres pour découper le texte et crées des segments à partir d'une quantité de données, Ainsi, le « StandardAnalyser » découpe le texte mot par mot, les met en minuscules puis retire ceux qui sont inutiles. On peut bien entendu créer nos propres combinaisons de filtres, et même en créer de nouveaux. Une façon plus

simple d'exécuter une telle tokenisation est d'utiliser la méthode « White-Space » : un terme se termine lorsqu'un espace blanc apparaît **[13]**. Toutefois, cela n'est pas utile si les termes fixes sont composés de plusieurs mots, à cet effet des dictionnaires sont également utilisés, et peuvent être implémentés directement dans le code Lucène.

Lors de l'analyse des données, dont la tokenisation fait partie, Lucène effectue également une normalisation pour transformer les termes en une forme standardisée. En outre, Lucène crée un ordre de tri. Ceci fonctionne via différents algorithmes tel que la mesure TF-IDF.

Une fois un analyseur créé, nous pouvons construire une instance d'IndexWriter et lui ajouter des documents. Le premier paramètre du constructeur d'IndexWriter désigne le nom du répertoire dans lequel sera enregistré l'index [13]. Le dernier paramètre permet de contrôler le mode de création de l'index. Lorsqu'il vaut true, un nouvel index sera créé, au risque de supprimer celui existant. Par contre en choisissant la valeur false, Lucène va ajouter les informations du nouvel index à celui existant.

Une fois l'IndexWriter créé, la compilation des documents est effectuée par l'intermédiaire de la classe Document, puis les segments sont placés dans l'index. Cette tâche est réalisée grâce à la méthode addDocument(). Un document se compose lui-même de champs, concrétisés par des instances de la classe Field, dans ce contexte on distingue trois sortes de champs à placer dans un document :

- Le type non indexé : réalisé par l'intermédiaire de la méthode utilitaire UnIndexed(), sert à ajouter des caractéristiques tel que l'URL à un document, et qui sont pas affectées par l'analyseur ;
- Le type non stocké : il est affecté par l'analyseur, mais la récupération de la valeur employée pour créer le champ est possible ;
- Le type indexé/stocké : il est en même temps indexé et accessible, il est du type Text().

Pour que les utilisateurs puissent rechercher quoi que ce soit, ils doivent alors entrer un terme de recherche dans une ligne de texte, les termes sont appelés « Query » dans le contexte de Lucène. Ce mot anglais pour requête indique que l'entrée ne doit pas seulement consister en un ou plusieurs mots, mais peut également contenir des modificateurs tels que AND et OR

ainsi que des caractères génériques[3], ensuite cette entrée est traduite en une demande de recherche concrète grâce à la classe « QueryParser » implémentée au sein de la bibliothèque du programme **[13]**. Cette classe fournit également aux développeurs des options de paramétrage, l'analyseur peut être configuré de manière à être exactement adapté aux besoins de l'utilisateur.

Ce que Lucène a apporté de complètement nouveau dès sa sortie est l'indexation incrémentale, avant Lucène, seule l'indexation par lots était possible. Alors que seuls des indexes complets[4] peuvent être implémentés, un index peut être mis à jour avec l'indexation incrémentale et des entrées individuelles peuvent être ajoutées ou supprimées. **[10]**

2.7 Conclusion :

Dans ce chapitre nous avons réalisé une étude approfondie sur le moteur de recherche Lucène. Dans un premier temps, nous avons défini la bibliothèque Lucène, dénombré ses fonctionnalités. Par la suite nous avons présenté son architecture notamment ses différents packages. Et enfin nous avons expliqué son fonctionnement global.

Le prochain chapitre portera sur les approches de la recherche d'information (les modèles de recherche d'information), notamment les approches implémentées dans Lucène.

[3] *Les caractères spéciaux utilisés pour remplacer des caractères inconnus (* et ?)*
[4] *Il est utilisé pour indexer les pages web.*

Chapitre 3

Les modèles de la recherche d’information

3.1 Introduction :

Le modèle de recherche d'information joue un rôle central dans la RI, il comprend la fonction de décision fondamentale qui permet d'associer à une requête, l'ensemble de documents pertinents à restituer. C'est lui qui détermine le comportement clé d'un SRI et, afin d'améliorer la pertinence de la recherche d'information dans ces SRI, plusieurs travaux ont été fait à divers niveaux .ainsi, il y a eu proposition de plusieurs modèles de recherches d'informations qui s'appuient sur des cadres théoriques différents ; théorie des ensembles, d'algèbre, de probabilités… etc.

3.2 Les modèles de recherche :

Le modèle de recherche d'information repose sur les représentations de la requête et des documents issues de l'indexation. Il permet de leurs donner une interprétation afin de déterminer les documents qui sont similaires à la requête.

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 les 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).

Il existe plusieurs modèles de recherche, ils sont généralement classés dans trois familles notamment, les modèles Booléens, les modèles vectoriels et les modèles probabilistes comme le présente la figure ci-dessous :

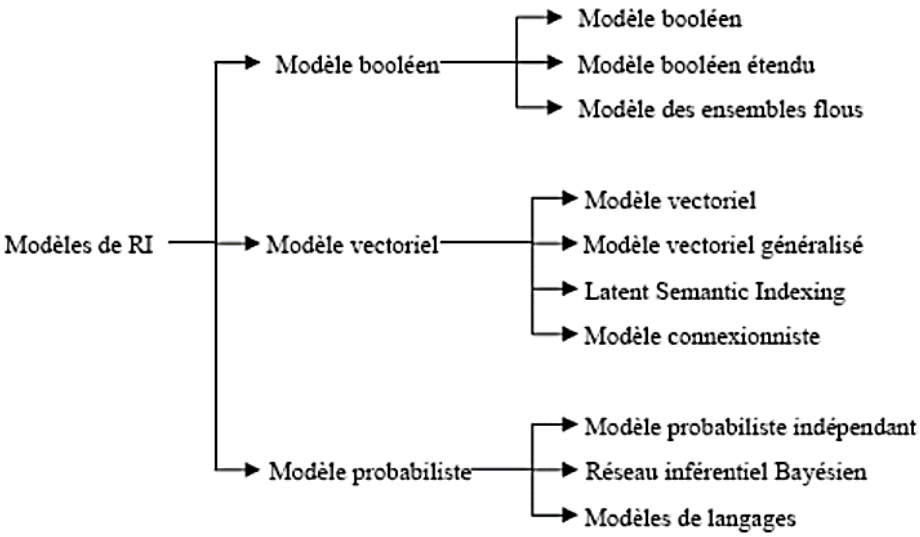

Figure 3.2.1: Classification des modèles de recherche . [14]

Dans ce qui suit, nous présentons ces principaux modèles utilisés.

3.2.1 Le modèle booléen :

Le modèle booléen est l'un des modèles classiques les plus utilisés dans les systèmes de recherche d'informations .Il en existe trois modèles notamment, le modèle booléen de base, le modèle booléen étendu et le modèle des ensembles flous.

3.2.1.1 Le modèle booléen de base :

Le modèle booléen de base est le premier modèle implémenté dans les SRI. Il est très simple et très rapide à mettre en œuvre. Ce modèle est basé sur la théorie des ensembles et l'algèbre de bool. Dans ce modèle, un document 'D' est représenté par un ensemble de mots clés, une requête 'Q' est représentée par une expression logique composée de mots-clés reliés par des opérateurs logiques (ET, OU, NON).

➢ Représentation des documents et des requêtes :

Dans ce modèle, un document est représenté par la conjonction de tous les termes d'indexation qui le composent. A titre d'exemple, si un document 'D' contient les termes "Recherche", "information", "modèle" et "web", alors il sera présenté comme suit :

D = (recherche ∧ information ∧ Modèle ∧ web)

Une requête est représentée par un ensemble de mots clés reliés par des opérateurs booléens (ET, OU et NON). A titre d'exemple, si un utilisateur cherche les documents contenant les termes "recherche" ou "information" et ne contenant pas le terme "Technologie", alors sa requête sera formulée par l'expression booléenne suivante :

Q = (recherche ∨ information ∧ ⌝ technologie)

Avec, ∧ représente l'opérateur ET, ∨ représente l'opérateur OU et ⌝ représente l'opérateur NON.

Formellement :

- Un terme est une requête;
- Si **q** est une requête alors **NON q** est une requête;
- Si **q1** et **q2** sont des requêtes alors **q1 ET q2** est une requête;
- Si **q1** et **q2** sont des requêtes alors **q1 OU q2** est une requête.

➢ Mesure de correspondance ou de similarité :

La fonction d'appariement *RSV(d, q)* qui détermine si un document 'd' répond à une requête 'q' est calculée comme suit:

- RSV (d, t_i)=1 si le terme $t_i \in$ d, sinon 0 ;
- RSV (d, *qi* ∧ qj)=RSV (d,q_i) × RSV(d, q_j) ;
- RSV (d, *qi* ∨*qj*)= RSV (d, q_i) + RSV (d, q_j) - RSV(d, q_i) × RSV(d, q_j);
- RSV(d, ⌝ qi)=1-RSV(d, qi) ;

Bien que ce modèle présente l'avantage d'être simple et facile à mettre en œuvre, il a toutefois ces inconvénients majeurs :

- Difficulté de formulation de bonnes requêtes par les utilisateurs (en effet le langage logique n'est pas à la portée de tout le monde) ;
- Les documents qui sont retournés aux utilisateurs ne sont pas ordonnés par ordre de pertinence (en effet, la RSV est binaire : un document est soit pertinent, soit non pertinent) [14].

3.2.1.2 Modèle booléen étendu «P-NORME » :

Pour surmonter les limites du modèle booléen, Salton, Fox et Wu ont présenté en 1983 **[15],** le modèle booléen étendu appelé aussi P-NORME.

- **Représentation des documents et des requêtes :**

La requête reste la même que celle du modèle booléen de base. Par contre, la fonction d'appariement utilise le poids des termes, qui sont préalablement normalisés pour être compris entre 0 et 1, afin de calculer un score représentant le degré de similarité entre un document et une requête.

- **Mesure de correspondance :**

Pour des requêtes à un ou deux termes :

RSV(d, t_i)= a_i.

RSV(d, ⅂ t_i)=1- RSV(d,t_i).

$$RSV(d, t_1 \vee t_2) = \sqrt{\frac{(1-RSV(d,t_1))^2+(1-RSV(d,t_2))^2}{2}}.$$

$$RSV(d, t_1 \wedge t_2) = \sqrt{\frac{RSV(d,t_1)^2+RSV(d,t_2)^2}{2}}.$$

t_i étant un terme quelconque et a_i son poids associé dans le document 'd'.

Les requêtes complexes contenant plusieurs termes sont traitées récursivement en appliquant les règles précédentes. Néanmoins, le modèle P-NORME propose une méthode permettant de calculer le score de similarité d'un document par rapport à la disjonction et la conjonction de plusieurs termes. Cette méthode se base sur les p-distances et est définie comme suit:

$$RSV(d, t_1 \text{ ET } t_2 \text{ET } \dots \text{ ET } t_m) = \sqrt{\frac{a_1^p + a_2^p + \cdots + a_m^p}{m}}$$

$$RSV(d, t_1 \text{ OU } t_2 \ \dots \text{OU } t_m) = 1 - \sqrt[p]{\frac{(1-a_1)^p + (1-a_2)^p + \cdots + (1-a_1)^p}{m}}$$

t_1, t_2 … t_m étant des termes quelconques et a_1, a_2 … a_m leurs poids associés dans le document 'd'. Le paramètre entier (p >=1) est indiqué au moment de la formulation de la requête.

L'avantage principal du modèle booléen étendu par rapport au modèle booléen de base est de pouvoir classer les documents suivant leur pertinence. Cependant, tout comme le modèle booléen de base, les requêtes restent complexes et difficilement formulées par l'utilisateur.

3.2.1.3 Modèle des ensembles flous :

Cette extension du modèle booléen standard **[15]** vise à tenir compte de la pondération des termes dans les documents.

- **Représentation des documents et des requêtes :**

Un document 'd' est représenté comme un ensemble de termes pondérés:

$D = \{. . . , (t_i, a_i), . . .\}$.

La requête reste une expression booléenne classique.

Avec : t_i étant un terme quelconque et a_i son poids associé dans le document 'd'.

- **Mesure de correspondance ou de similarité :**

L'évaluation de la pertinence d'un document pour une requête peut prendre plusieurs formes :

- **Evaluation de Zadeh (1965):**

- $RSV(d,t_i)=a_i$
- $RSV(d,q_i \text{ AND } q_j)= MIN (RSV(d,q_i), RSV(d,q_i))$
- $RSV(d,q_i \text{ OR } q_j)= MAX(RSV(d,q_i), RSV(d,q_i))$
- $RSV(d, \text{NOT } q_i)= 1- RSV(d,q_i)$

L'avantage principal de ce modèle est d'offrir la possibilité de classer les documents par ordre de pertinence. Cependant, il présente certains inconvénients. Notamment le fait qu'une évaluation d'une conjonction favorise le petit poids et celle d'une disjonction favorise le grand poids. Ceci fait qu'un système utilisant ce modèle peut retourner des résultats inattendus pour certaines requêtes.

3.2.2 Le modèle vectoriel :

Ce modèle introduit par Gérard Salton [16]. est basé sur la théorie de l'algèbre et plus précisément sur le calcul vectoriel **[3],** sur la détermination de l'espace des documents pertinents, dans lequel la valeur de chaque document est déterminée en fonction de son degré de relation avec la requête. on y retrouve ; le modèle vectoriel de base, le modèle vectoriel généraliste, LSI (Latent Semantic Indexation) et modèle connexionniste.

3.2.2.1 Le modèle vectoriel de base :

Le modèle vectoriel de base fait partie des premiers modèles utilisés en recherche d'information .Il a été proposé au début des années 70 par Gérard Salton dans le cadre du système SMART (Salton's Magical Automatic Retriever of Text). Il se base sur l'idée que les documents les plus pertinents sont ceux les plus proches des requêtes (c'est-à-dire qui contiennent les mêmes termes). Ce modèle utilise une représentation géométrique, les documents et les requêtes sont représentées sous forme de vecteurs de poids dans un espace à n-dimensions où chaque dimension représente un terme d'index.

- **Représentation des documents et des requêtes :**

La figure 2 illustre de façon graphique la représentation de deux documents $\overrightarrow{d_1}$et $\overrightarrow{d_2}$ et d'une requête $\vec{q}$ dans un espace associé à deux termes.

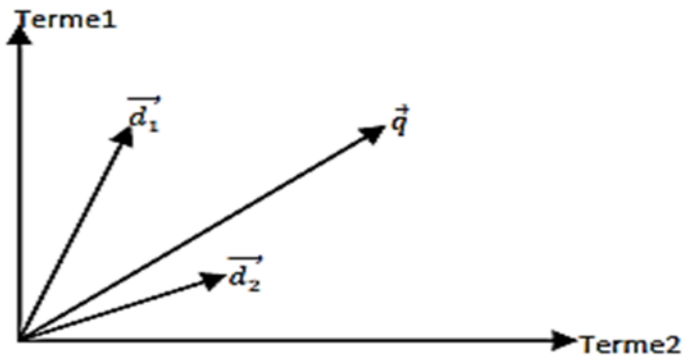

Figure 3.2.2: Exemple d'une représentation du modèle vectoriel

Dans un espace d'information à n-terme, Le vecteur associé à un document est représenté comme suit :

$$\overrightarrow{d_j} = (w_{1j}, w_{2j}, \dots, w_{nj})$$

Tel que : w_{ij} correspond au poids d'un terme dans le document d_j.

Le vecteur associé à la requête est défini par :

$$\vec{q} = (w_{1q}, w_{2q}, \dots, w_{iq})$$

Tel que : w_{iq} correspond au poids d'un terme dans la requête q

- **Mesure de similarité ou correspondance :**

La correspondance d'un document avec la requête utilisateur est définie par la similarité de leurs vecteurs associés calculée à partir de l'une des mesures de similarité vectorielle suivantes :

— Le produit scalaire :

$$\mathrm{RSV}(q, d_j) = \sum_{i=1}^{n} (w_{iq} * w_{ij})$$

— La mesure de Jaccard :

$$\mathrm{RSV}(q, d_j) = \frac{\sum_{i=1}^{n} (w_{iq} * w_{ij})}{\sum_{i=1}^{n} {w_{iq}}^2 + \sum_{i=1}^{n} {w_{ij}}^2 - \sum_{i=1}^{n} (w_{iq} * w_{ij})}$$

— La mesure de Cosinus :

$$\mathrm{RSV}(q, d_j) = \frac{\vec{q} * \overrightarrow{d_1}}{\|\overrightarrow{q}\| * \|\overrightarrow{d_1}\|} = \frac{\sum_{i=1}^{n} (w_{iq} * w_{ij})}{\sqrt{\sum_{i=1}^{n} {w_{iq}}^2 * \sum_{i=1}^{n} {w_{ij}}^2}}$$

— La mesure de Dice :

$$\mathrm{RSV}(q, d_j) = \frac{2 * \sum_{i=1}^{n} (w_{iq} * w_{ij})}{\sum_{i=1}^{n} {w_{iq}}^2 + \sum_{i=1}^{n} {w_{ij}}^2}$$

Où $W_{i,j}$ est le poids du terme ' i ' dans le document ' j' et $W_{i,q}$ est le poids du terme i dans la requête 'q'.

Le modèle vectoriel de base est l'un des modèles les plus influents, les plus étudiés et les mieux acceptés en domaine de RI. Il a trouvé son succès dans sa simplicité conceptuelle et de

mise en œuvre. En fait, les mesures de similarité employées par ce modèle permettent à l'utilisateur d'avoir une liste triée des documents pertinents, en plaçant en tête les documents jugés les plus similaires à la requête. Cependant, ce modèle a plusieurs inconvénients. A titre d'exemple, il ne permet pas de modéliser les dépendances entre les termes d'indexation, chacun des termes est considéré comme indépendant des autres [14].

3.2.2.2 Le modèle vectoriel généralisé (Generalized Vector Space Model) :

Afin de résoudre le problème de l'indépendance des termes d'indexation posé par le modèle précédent, Wong **[17]** a proposé une nouvelle représentation de documents et requêtes en considérant qu'il peut exister des dépendances entre les termes d'indexation. De ce fait, on ne peut plus représenter les documents et les requêtes dans un même espace que celui du modèle vectoriel de base où les axes représentent les termes d'indexation, car ceux-ci sont orthogonaux et impliquent l'existence d'une indépendance entre les termes.

Pour remédier à ce problème, un nouvel espace est défini où les axes ne représentent plus les termes d'indexation, mais un ensemble de descripteurs virtuels, construit à partir des termes d'indexation en tenant compte de leurs cooccurrences dans les documents.

- **Représentation des documents et des requêtes :**

Un document est représenté dans cet espace par un vecteur défini comme suit:

$$\vec{d} = \sum_{i=1}^{m} w_{d,i} * \vec{x_i}$$

Où :

m : est le nombre de descripteurs virtuels présents dans cet espace ;

$\vec{x_i}$: est un vecteur de base, associé au $i^{ème}$ descripteur virtuel ;

$W_{d,i}$ correspond au degré de pertinence dans le document $\vec{d}$ du descripteur virtuel défini par $\vec{x_i}$.

Le vecteur associé à la requête est défini de la même manière.

➢ **Mesure de Correspondance ou de similarité :**

Le degré de similarité entre une requête et un document est calculé en appliquant les mêmes formules que celles utilisées dans le modèle vectoriel de base mais en utilisant les nouveaux vecteurs requête et document ainsi définis précédemment .De ce fait, le calcul du degré de similarité est plus couteux dans ce modèle, comparé au modèle vectoriel de base, mais il introduit la notion de dépendance entre les termes.

3.2.2.3 Le modèle connexionniste :

Les systèmes de recherche d'informations basés sur le modèle connexionniste sont fondés sur les réseaux de neurones **[18]** ; ses premiers modèles datent des débuts de l'intelligence artificielle avec le célèbre perceptron de Rosenblatt en 1962 **[19]**. En s'inspirant de la représentation et le traitement de l'information dans le système nerveux humain, ce modèle partitionne du même le corpus en une couche d'entrée est une couche de sortie. Par ailleurs, un réseau de neurones est composé de nœuds et de liens, les nœuds représentent les entrées et les sorties, tandis que les liens se caractérisent par un poids traduisant le degré d'interconnexion entre nœuds **[19]**. La représentation graphique des neurones est la suivante :

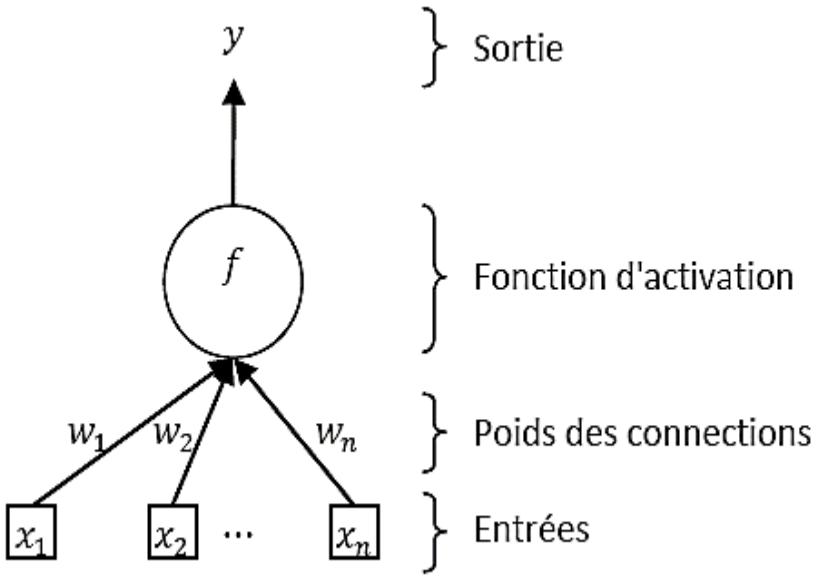

Figure 3.2.3: Représentation graphique des neurones.

- **Mécanisme de fonctionnement :**

Un réseau connexionniste est constitué d'unités dites neurones formels qui représentent les objets de la recherche d'information, elles sont reliées entre elles grâce à des connexions, à chaque connexion est associé un poids qui caractérise l'influence de l'unité source de la connexion sur l'unité cible. Dans ce modèle, le réseau est construit à partir des représentations des contenues des requêtes et de document sous forme de couches répartis généralement dans ce sens : requête - termes – document **[20].** Le mécanisme de recherche d'information basé sur ce modèle doit automatiquement passer par une phase d'apprentissage où les résultats sont retournés par les poids des connections entre les neurones. En effet, la requête active la couche des termes, ce qui engendre l'activation de la couche de document à travers les connexions du réseau. Ensuite les résultats sont présentés à l'utilisateur selon le niveau d'activation des neurones document. Par conséquent, la pertinence de document est alors mesurée grâce à leur niveau d'activation.

3.2.2.4 Le modèle Latent Semantic Indexing :

Le modèle LSI est une extension du modèle vectoriel **[21]**, il a été proposé comme solution au problème traditionnelle basée uniquement sur les mots **[22]**. Il se caractérise par la prise en compte de la structure sémantique, et la transformation de la représentation traditionnelle par mots-clés en une représentation plus conceptuelle qui vise à favoriser le rapprochement de documents et requêtes sémantiquement.

- **Représentation des documents et des requêtes :**

Dans ce modèle, le document et la requête sont représentés chacun comme un vecteur de mots-clés, ensuite on crée un nouvel espace vectoriel en se basant sur la décomposition en valeurs singulières (SVD) pour identifier la relation entre eux.

Dans une collection de 'n' documents d_1, d_2,….. d_n , contenant 'm' termes d'indexation , la matrice d'occurrence $M=(w_{i,j})$ avec i=1..m et j=1..n est une matrice de dimension m×n où chaque élément $w_{i,j}$ représente le poids associé au terme 't_i' dans le document 'd_j'.

$$M=\begin{pmatrix} w_{1,1} & \cdots & w_{1,n} \\ \vdots & \ddots & \vdots \\ w_{m,1} & \cdots & w_{m,n} \end{pmatrix} = (\overrightarrow{d_1}. \ldots \overrightarrow{d_2})$$

Cette approche consiste à réduire la dimension de l'espace représentant d'une manière à garder le plus d'information possible. Par ailleurs, la matrice d'occurrence est décomposée en valeurs singulières et donne naissance à une nouvelle matrice F de dimension r×r avec r=min(m,n)

$$F=U.\ M.V^t$$

Avec :

- F la matrice d'occurrence;
- U et V sont des matrices unitaires de dimension m×r et n×r respectivement, contenant un ensemble de vecteurs orthonormés (c.-à-d. $U.U^t=1$ et $V.V^t=1$);
- V^t est la matrice transposée de la matrice V, de dimension r×n ;
- F est une matrice diagonale de dimension r×r contentant les valeurs singulières de M.

La matrice F est ensuite ordonnée de façon décroissante et réduite à une dimension k, avec k < r, en considérant que les k valeurs les plus grandes de F suffisent pour représenter l'information contenue dans M :

$$F=U.M.V^t\ =U'.\ M'.V^{t'}\ =\ F'$$

Avec :

- F' la nouvelle matrice (réduite) en utilisant le coefficient k;
- U' et V' sont des matrices unitaires de dimension m×k et n×k respectivement, construite à partir de U et V ;
- F' est la meilleure approximation de rang k de la matrice F.

En effet la nouvelle représentation des documents sera une nouvelle matrice de dimension n × k, qui dépend d'un nombre limité de termes, car les termes identiques sémantiquement parlant, sont considérés comme une seule entité. Ainsi V contient n lignes correspondant chacune aux coordonnées d'un vecteur document. Et la représentation du vecteur document 'd'est la suivante :

$$d=d^T.U.M^t$$

Du même, nous définissons le vecteur requête comme suite :

$$q=q^T.U.M^t$$

- **Mesure de similarité entre document et requête :**

Dans ce modèle les lignes et les colonnes de la matrice ainsi que la requête sont mappés dans l'espace k dimensionnel LSI. Maintenant il reste qu'à calculer le degré de similarité du vecteur requête à tous les documents de la collection suivant cette méthode :

$$sim(q,d)=sim(q^T U_k S_k^{-1}, d^T U_K S_k^{-1})$$

Pour ce faire, on utilise les mêmes formules de correspondance définies précédemment dans le modèle vectoriel de base.

Pour les corpus de petite ou moyenne taille, ce modèle a montré des performances très intéressantes par rapport au modèle vectoriel de base, pas seulement dans la prise en compte de la structure sémantique et le filtrage d'information mais également dans la recherche documentaire multilingue. Cependant, la mise en œuvre de ce modèle nécessite des traitements plus coûteux que le modèle vectoriel de base. En outre, la réduction des dimensions de l'espace de représentation final est difficile à interpréter ; il n'y a pas de règles théoriques générales permettant de trouver une valeur optimale pour l'indice k. Par conséquent, un mauvais choix peut engendrer une mauvaise approximation (k très petit) ou une décomposition sans intérêt (k très grand).

3.2.3 Le modèle probabiliste :

La recherche d'information est un processus incertain et imprécis à cause de l'imprécision dans l'expression des besoins de l'utilisateur (la requête) et l'incertitude dans la représentation des informations. Du coup, la théorie des probabilités semble très adéquate pour prendre en compte les problèmes liés à ce processus **[23].**

3.2.3.1 Définition :

Proposé par Maron et Kuhns au début des années 1960 **[24]**, l'approche probabiliste tente de résoudre le problème de la RI en utilisant un modèle mathématique fondé sur la théorie de la probabilité. Le modèle probabiliste consiste à calculer la pertinence d'un document à une requête en fonction de pertinences connues pour d'autres documents, ce calcul se fait en estimant la pertinence de chaque index pour un document en utilisant le théorème de Bayes et une règle de décision.

Il existe différentes manières de voir une approche probabiliste dans le domaine de la recherche d'information :

• Approche classique : probabilité d'avoir l'événement Pertinent sachant un document et une requête.

• Approche par réseaux d'inférence : A partir du contenu d'un document, On détermine la probabilité pour que la requête soit vraie.

• Approche par modèle de langue : Déterminer la probabilité qu'une requête posée soit générée à partir d'un document.

Dans ce qui suit, nous allons commencer par faire un rappel sur les notions fondamentales des probabilités, ensuite nous allons présenter certaines approches probabilistes en détail.

3.2.3.2 Rappels probabilistes :

Soit P(e) la probabilité d'un événement 'e' :

$0 < P(e) < 1$

P("pile") = P("face")=1/2

$\Sigma P(e) = 1$ (tous les événements possibles)

$\neg P(e) = 1 - P(e)$

$P(A \cup B) = P(A) + P(B) - P(A \cap B)$

L'estimation des probabilités revient à compter le nombre de cas favorable sur le nombre de cas total.

- **Probabilité conditionnelle P(A|B) :**

C'est la probabilité conditionnelle qui signifie la probabilité de 'A' sachant que 'B' est réalisé, à noter que 'A' et 'B' sont dépendants dans l'ensemble 'E', on a :

$$P(A|B) = P(A \cap B) / P(B)$$

Avec: $P(A \cap B) = |A \cap B| / |E|$

$P(B) = |B| / |E|$

Ce qui donne : $P(A,B) = P(A \cap B) = P(A|B) \times P(B) = P(B|A) \times P(A)$

D'où on obtient la règle de Bayes : $P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$

- **Variable aléatoire :**

C'est une fonction qui associe à chaque résultat d'une expérience aléatoire un nombre (un réel) $X : \Omega \rightarrow R$

$\omega \rightarrow X(\omega)$

On appelle univers associé à une expérience aléatoire l'ensemble Ω de tous les résultats possibles R de cette expérience.Une variable aléatoire peut être discrète (ensemble Ω dénombrable) ou continue

- **Loi de probabilité pour une variable aléatoire :**

Elle décrit la probabilité de chaque valeur x_i d'une variable aléatoire, on note:

$$p_i=P(X=x_i) \text{ avec } 0 \leq p_i \leq 1 \text{ et } \Sigma pi=1$$

- **Loi uniforme** : la variable aléatoire prend les valeurs X={1,2,…,n}

$$P(X= k) = \frac{1}{n}$$

- **Loi de Bernoulli** : X={0,1}

$$P(X=1)= p$$
$$P(X=0)= 1- p$$
$$P(X= x)= p^x (1-p)^{1-x}$$

Apres avoir présenté les différentes lois utilisées dans le modèle probabiliste, on va passer à la description de ses approches.

3.2.3.3 Le modèle probabiliste de base :

Ce modèle consiste à retourner pour chaque requête une liste de documents dans un ordre décroissant basé sur la probabilité de pertinence d'un document vis-à-vis d'une requête en s'appuyant sur le principe d'ordonnancement (tri) probabiliste (Probability Ranking Principle). Il a était proposé par Robertson en 1977 qui considère chaque paire (d, q) comme la réalisation d'une expérience aléatoire qui consiste à tirer simultanément le document et la requête de deux ensembles prédéfinis de documents et de requêtes. À chaque tirage (d, q) on associe une variable aléatoire binaire égale à 1 si 'd'est pertinent pour 'q' et 0, sinon. Le but de ce modèle est de modéliser l'expérience aléatoire et d'estimer les paramètres du modèle.

➢ **Définition :**

On définit P(R|d) comme la probabilité pour que le document 'i' fasse partie de l'ensemble des documents pertinents à la requête 'q', et P(NR|d) comme la probabilité pour que le document 'i' fasse partie de l'ensemble des documents non pertinents à la requête 'q'.

Pour chaque requête 'q' on a :

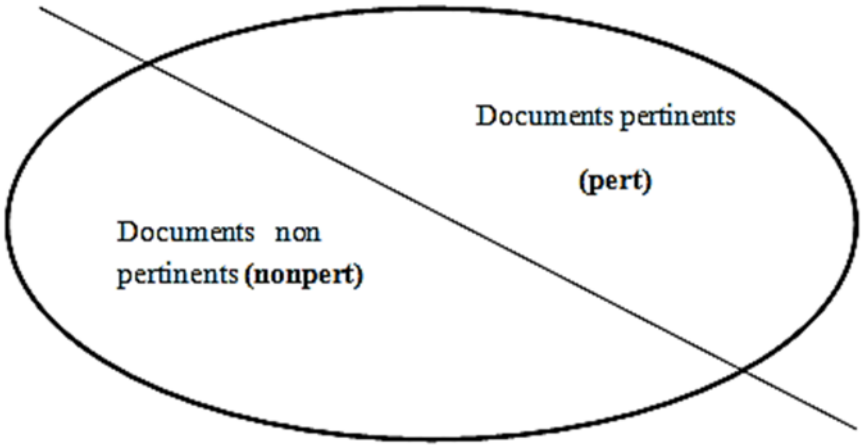

Figure 3.2.4: Corpus

Avec : Corpus = pert ∪ nonpert

pert ∩ nonpert = ∅

Ce modèle estime la probabilité qu'un document appartient à la classe des documents pertinents (ou non pertinents) en s'appuyant sur deux probabilités conditionnelles : Un document 'd' est sélectionné si la probabilité qu'il soit pertinent pour une requête 'q', notée P(R/d), est supérieure à la probabilité qu'il soit non pertinent, notée P(NR/d).

Le score d'appariement entre le document 'd' et la requête 'q', noté RSV(d,q) après simplification est donnée par :

$$\mathrm{RSV}(d,q) = \frac{P(R/d)}{P(NR/d)}$$

En supposant l'indépendance des variables documents « pertinents » et « non pertinents », la fonction de recherche peut être obtenue en utilisant la formule de Bayes :

$$\mathrm{RSV}(d,q) = \frac{P(R/d)}{P(NR/d)} \cong \frac{P(d/R)}{P(Nd/R)}$$

Ainsi le document 'd' répond à la requête si :

$$\frac{P(d/R)}{P(Nd/R)} > 1$$

Avec ce modèle, on sélectionne les documents ayant à la fois une forte probabilité d'être pertinents et une faible probabilité d'être non pertinents à la requête.

Dans ce modèle, on ne prend en compte que l'absence ou la présence des termes dans les documents et dans la requête. Par conséquent, les termes ne sont pas pondérés mais prennent seulement des valeurs 0 (absent) ou 1 (présent) **[25]**. Cela a donné naissance à plusieurs méthodes pour estimer ces différentes probabilités telles que le modèle d'indépendance binaire et le modèle de poisson. Dans ce qui suit nous décrivons particulièrement le modèle d'indépendance binaire connu sous le modèle de BIR (Binary Independance Retrieval).

3.2.3.3.1 Le modèle de BIR :

- **Représentation du document et de la requête :**

Dans ce modèle, une variable document 'd' est représentée par un ensemble d'événements t_i : d ($t_1 = x_1$, $t_2 = x_2$, ….$t_n = x_n$).

Un évènement t_i dénote la présence ($x_i = 1$) ou l'absence ($x_i = 0$) d'un terme dans un document

- **Mesure de similarité :**

Considérons chaque document comme une liste de termes, on obtient :

$$\mathrm{RSV}(q,d) = \frac{P(d/R)}{P(d/NR)} = \frac{P((t_1,t_2 \ldots\ldots t_n)/R)}{P((t_1,t_2 \ldots\ldots t_n)/NR)}$$

En s'appuyant sur l'hypothèse d'indépendance des termes :

$$\mathrm{RSV}(q,d) = \prod_{i=1}^{n} \frac{P(t_i/R)}{P(t_i/NR)}$$

Dans ce cas, t_i peut être vu comme une variable aléatoire et la distribution des termes suit une loi de Bernoulli :

On a $x_i = \begin{cases} 1 \\ 0 \end{cases}$ alors $p_i = P(t_i = 1|R)$ entraine $1- p_i = P(t_i = 0|R)$

et $q_i = P(t_i = 1|NR)$ entraine $1- q_i = P(t_i = 0|NR)$

D'où : $$P(d/R) = \frac{\prod_{i=1}^{n} P(ti=xi/R)}{\prod_{i=1}^{n} P(ti=xi/NR)} = \frac{\prod_{i=1}^{n} pi^{xi}\,(1-pi)^{1-xi}}{\prod_{i=1}^{n} qi^{xi}\,(1-qi)^{1-xi}}$$

En se ramenant à la fonction log, la fonction RSV s'écrit alors :

$$RSV(q,d) = \log\frac{P\left(\frac{d}{R}\right)}{P\left(\frac{d}{NR}\right)} = \log\frac{\prod_{i=1}^{n} pi^{xi}\,(1-pi)^{1-xi}}{\prod_{i=1}^{n} qi^{xi}\,(1-qi)^{1-xi}} = \sum_{i,xi}\log\frac{pi\,(1-qi)}{qi\,(1-pi)}$$

- **Estimation de 'p'et 'q' sur un ensemble de requêtes prédéfinis [23] :**

Table 3.2.1: Estimation de 'p' et 'q'

	Pertinent	Non pertinent	Total
Terme t_i présent	r_i	$n_i - r_i$	n_i
Terme t_i absent	$R- r_i$	$N- n_i - (R- r_i)$	$N- n_i$
Total	R	N-R	N

Avec :

R : nombre de documents pertinents pour q (contenant t_i) ;

N: cardinalité du corpus ;

r_i: nombre de documents pertinents contenant t_i ;

n_i : nombre de documents contenant le terme t_i ;

- Pertinent et non pertinent par rapport aux documents D ;
- Terme t_i (présent ou non présent) par rapport au document dont on veut mesurer la pertinence par rapport aux documents D.

Et on aura : $p_i = \frac{r}{R}$ $q_i = \frac{n-r}{N-R}$

En remplaçant dans la formule précédente, on obtient la formule globale suivante :

$$RSV(q,d) = \sum \log \frac{r/(R-r)}{(n-r)/(N-n-R+r)}$$

Pour éviter les valeurs nulles, on procède au lissage avec la formule de Robertson Spark Jones [24] qui donne :

$$RSV(q,d) = \sum \log \frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)}$$

Pour conclure, Le modèle de BIR est une formalisation puissante qui modélise la notion de pertinence d'une manière explicite. En revanche il est difficile d'estimer les probabilités sans données d'apprentissage, et contrairement aux autres approches de la recherche d'information, la fréquence des termes n'est pas prise en compte par ce modèle.

3.2.3.4 Le modèle probabiliste de langue :

L'approche de modélisation de langue a été proposée comme une nouvelle alternative au modèle vectoriel traditionnel et aux autres modèles probabilistes **[22].**Contrairement aux approches classiques précédentes qui s'appuient sur l'évaluation de similarité, ce modèle estime que la pertinence d'un document par rapport à une requête est calculée en se basant sur la probabilité que la requête puisse être générée à partir du document.

Il s'agit d'un modèle statistique basé sur des corpus, au lieu des connaissances préétablies, il a connu de grands succès dans la linguistique informatique **[26].**Ces méthodes tentent de capter, d'une manière statistique, les régularités d'une langue en observant des phrases dans un corpus d'entraînement. En effet, elles sont intégrées dans plusieurs application de traitement automatique de langue tels que l'étiquetage des catégories syntaxiques (part-of-speech tagging), la reconnaissance de parole (speech recognition)**[27]**, et même en traduction automatique**[28]** (machine translation). Ce succès est non seulement dû à l'avancement des méthodes utilisées, mais aussi à la disponibilité de plus en plus des corpus d'entraînement de différentes sortes**[29]**.

➢ Définition :

Le modèle de langue est une méthode basée sur des corpus, elle s'appuie sur des techniques statistiques pour estimer à la fois le modèle du document et le score du document pour une requête.

Le principe de ce modèle consiste à construire un modèle de langue pour chaque document soit M_d puis de calculer la probabilité qu'une requête 'q' puisse être générée par le modèle de langue du document, soit $P(q/M_d)$ **[3]**, cette probabilité est exprimée ainsi :

$$RSV(d,q)= P(q/M_d)= P(q=(t_1,t_2\ldots\ldots t_n)/Md)= \prod_{j=1}^{m} P(t_j/d)$$

➢ Représentation des documents et des requêtes :

Dans ce modèle, chaque document 'd' est décrit par un modèle de langue M_d en considérant que 'd'est une séquence ' s' composée de mots : m_1, m_2, …, m_i et que la requête 'q' est également une suite indépendante de mots : t_1, t_2…. t_n

Pour définir un modèle de langue, on doit d'abord définir la taille des séquences générées par le modèle :

- Séquence d'un mot → modèle unigram ;
- Séquence de deux mots → modèle bigram ;
- Séquence de n mots → modèle de n-gram.

Le modèle unigram est le plus utilisé en RI, les textes sont donc générés à partir de mots simples de manière indépendante les uns des autres. Si m_1, m_2, ..m_n est le vocabulaire généré par le modèle, alors l'estimation du modèle reviens à calculer la probabilité de chaque séquence 's' générée, notée P(s|M)

Avec $P(s_1)+P(s_2)+\ldots P(s_n)=1$

Soit 's' une observation (un texte) de n mots $s=m_1, m_2\ldots..m_n$:

Unigram: $P(s) = P(m_1, m_2\ldots..m_n)= \prod_{i=1}^{n} P(m_i)$

Bigram : $P(s) = \prod_{i=1}^{n} P(m_i/m_{i-1}) = \prod_{i=1}^{l} \frac{P(m_{i-1}m_i)}{P(m_{i-1})}$

Trigram: $P(s) = \prod_{i=1}^{n} P(m_i/m_{i-2}\, m_{i-1}) = \prod_{i=1}^{n} \frac{P(m_{i-2}\, m_{i-1}\, m_i)}{P(m_{i-2}\, m_{i-1})}$

Ce que l'on doit estimer sont les probabilités $P(m_i)$, $P(m_{i-1}m)$ et $P(m_{i-2}m_{i-1}\, m_i)$. Cependant, il est difficile d'estimer ces probabilités pour une langue dans l'absolue. L'estimation ne peut

se faire que par rapport à un corpus de textes, et si le corpus est suffisamment grand, on peut faire l'hypothèse qu'il reflète la langue en général **[29]**. De cette façon, on peut estimer approximativement le modèle de langue pour tout le corpus 'C'. Selon les fréquences d'occurrence d'un n-gramme notée tf , sa probabilité $P(m_i/C)$ peut être estimée comme suit :

$$P(m_i/C) = \frac{tf}{\sum i \in C\ tf_i}$$

Où : tf indique la fréquence d'occurrence d'un n-gramme dans le corpus C ;

$\sum i \in C\ tf_i$ le nombre total d'occurrences du mots.

À noter que le n-gramme et tf possède la même longueur.

Plus le corpus utilisé pour ces estimations est grand, plus la possibilité d'obtenir des estimations justes augmente **[29].** En outre, quelle que soit sa taille, il y a toujours des mots qui n'apparaissent pas dans le corpus d'entraînement, et le modèle leur assigne une probabilité nulle. Par conséquent, les phrases contenant un mot non rencontré dans le corpus auront également des probabilités nulles. Pour remédier à cela, on utilise des techniques de lissage qui consiste à attribuer une probabilité non-nulle á ces éléments. Parmi ces techniques, le lissage de Laplace qui consiste à ajouter la fréquence 1 à tous les n-grammes, et donc sa probabilité est estimée comme suit:

$$P(m_i/C) = \frac{tf+1}{\sum i \in C\ \ tf_i+1}$$

3.3 Les approches de recherches implémentées dans Lucène :

Pour faciliter la tâche d'implémentation des approches précédentes, Lucène a mis à la disposition des chercheurs une multitude de similarités prêtes à l'emploi. Par ailleurs une classe est consacrée dans chaque version de l'api pour accueillir l'ensemble des similarités implémentées. L'avantage est que cette classe est mise à jour à chaque nouvelle version, ce qui engendre l'amélioration des algorithmes utilisés et la naissance de nouvelles similarités. Pour accomplir notre travail, nous avons choisis de travailler avec la version Lucène 8.6.3 suite à sa récence, ce qui nous a permis de bénéficier des dernières fonctionnalités de l'api.

Dans ce qui suit, nous définissons les différentes similarités que nous avons utilisées pour réaliser notre étude comparative.

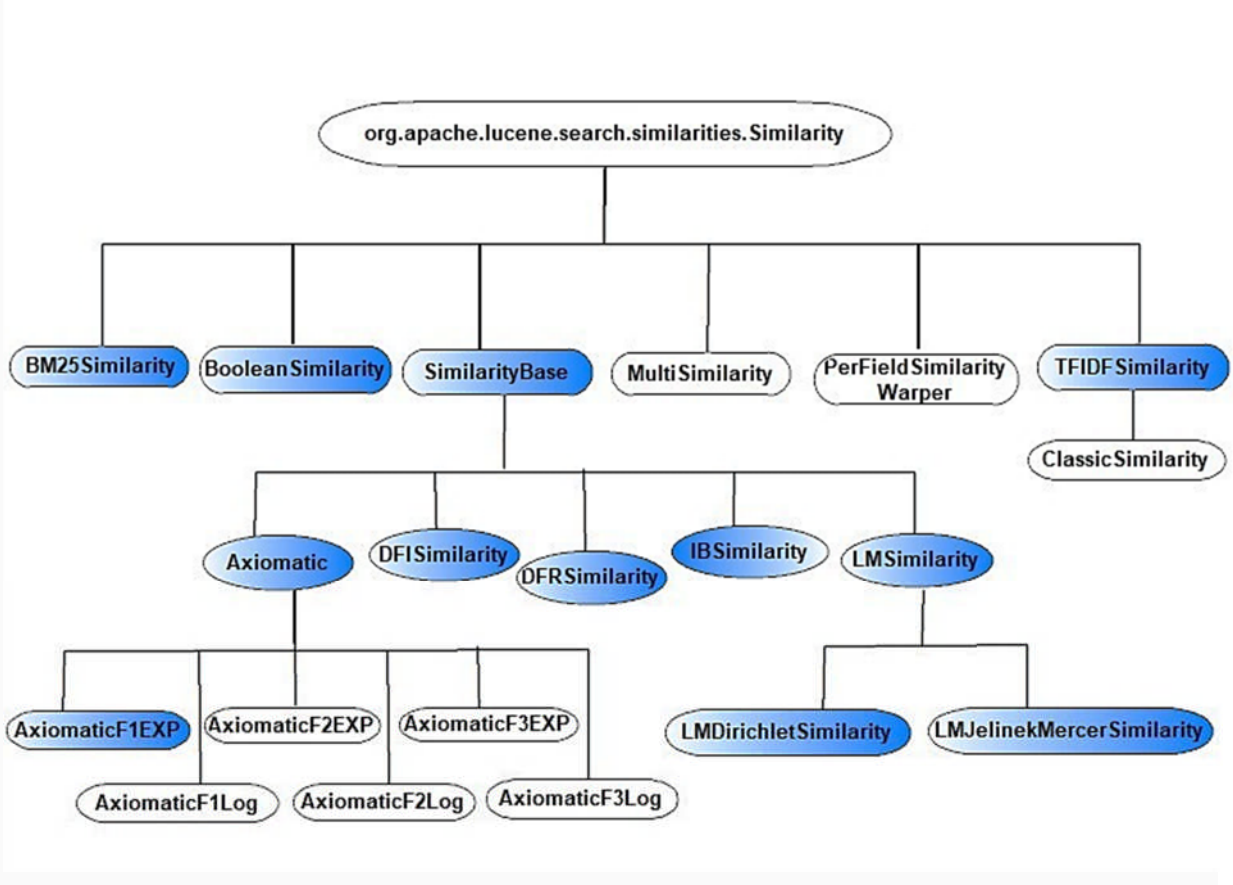

Figure 3.3.1: les différentes similarités implémentées dans Lucène 8.6.3

3.3.1 Le modèle TFIDF :

Implémenté sous le nom de classique similarité dans Lucène, ce modèle est une combinaison du modèle booléen (BM) avec le modèle spatial vectoriel (VSM), les documents approuvés par BM sont notés par VSM **[30].** En d'autre terme, pour un terme 't' et un document 'd' donné (ou une requête), Tf(t,d) varie et augmente proportionnellement suivant le nombre d'occurrences du terme 't' dans 'd', du même, idf (t) varie avec l'inverse du nombre de documents d'index contenant le terme 't' **[30]**.

Le score VSM du document 'd' pour la requête 'q' est donné comme suit :

$$\text{Sim cos}(d,q) = \frac{V(q).V(d)}{|V(q)|\,|V(d)|}$$

Où V (q).V (d) le produit scalaire des vecteurs pondérés ;

|V(q)| et |V(d)| sont leurs normes euclidiennes.

3.3.2 Okapi BM25 :

Cette pondération a initialement été proposée comme modèle de similarité dans un cadre probabiliste en 1976 par Robertson et Jones **[31]**, elle constitue un algorithme de pertinence probabiliste, ce qui signifie que le score peut être considéré comme la probabilité qu'un document donné corresponde à la requête.

La pondération Okapi[1] est devenue une référence grâce aux très bons résultats qu'elle permet d'obtenir sur de nombreuses tâches de la RI.

Étant donné une requête Q, contenant des termes $q_1,q_2,....,q_n$ le score BM25 d'un document d est:

$$\text{score}(Q,d) = \sum_{i=1}^{n} \text{idf}(q_i).\frac{f(q_i,d)(k_1+1)}{f(q_i,d)+k_1.(1-b)+b.\frac{|d|}{\text{avgd}}}$$

Où : $f(q_i,d)$ est la fréquence du terme q_i dans le document d ;

1 La formule de cette pondération s'appelle en réalité BM-25, mais est souvent appelée Okapi du nom du premier système l'ayant implémenté.

| d | : langueur de document en nombre de termes ;

Avgd : la longueur moyenne des documents dans la collection ;

Idf(q_i) est la fréquence inverse de document.

K_1 et b sont des paramètres libres pouvant être optimisés selon les cas d'utilisation mais qui, en l'absence de toute optimisation sont usuellement fixés à k1 ∈ [1.2 ,2.0] et b=0.75

Sur la version Lucène 8.6.3, cette approche peut être utilisée avec des valeurs de paramètres par défaut (k1 = 1.2 b = 0.75) ou avec des valeurs fournit par l'utilisateur.

3.3.3 La similarité booléenne :

Est une simple similarité qui donne aux termes un score égal à leur augmentation de requête **[30]**autrement dit, augmenter le score du terme ayant plus d'importance dans la requête. Pour ce faire, un facteur multiplicatif est appliqué aux scores des termes. Il prend une valeur d'amplification par défaut égale à 1,et pour faire amplifier la requête il suffit de choisir une valeur supérieure à 1, tandis que pour diminuer l'importance d'un terme on utilise une valeur comprise entre 0 et 1.

3.3.4 Le modèle DFR :

Divergence From Randomness (DFR), en français divergence par rapport au hasard, est parmi les modèles de la recherche d'information appartenant à l'approche probabiliste, il a été proposé par Amati et Rijsbergen en 2002 **[32]**, Son but principal est de tester la quantité d'informations contenues dans les documents en s'appuyant sur les fondements du modèle d'indexation qui suit une loi de Poisson de Harter puisque les occurrences d'un mot dans un document sont distribuées de manière aléatoire.

Ce modèle calcule les poids des termes en mesurant la divergence entre une distribution de terme produite par un processus aléatoire (au sein du document) et la distribution de terme réelle (au sein du corpus**)** **[32]**. Son principe est le suivant, plus la divergence de la fréquence-terme intra-document par rapport à sa fréquence au sein de la collection est importante, plus l'information est portée par le mot 't' dans le document 'd'. En d'autres termes, le poids est inversement proportionnel à la probabilité de terme-fréquence dans le document 'd' obtenue par un modèle M d'aléatoire.

Le processus du DFR est mis en place en commençant par sélectionner un modèle d'aléatoire de base, puis appliquer une première normalisation et enfin normaliser les termes

fréquences. Quant à Lucène, ces trois composants sont représentés respectivement par les classes BasicModel, AfterEffect et Normalization **[30].** En effet, pour implémenter cette similarité sur Lucène, le choix des trois composants de DFR est nécessaire.

3.3.5 DFI Similarity (Divergence From Independence):

Le modèle DFI est étroitement lié à la divergence du modèle aléatoire (DFR),cependant ils sont différents en ce sens qu'en DFR, on suppose que les termes importants d'un document donné sont les mots dont les fréquences diffèrent des fréquences suggérées par un modèle aléatoire de base, tel que Poisson, Hyper-Géométrique, Bose-Einstein etc., alors que dans DFI, il suppose que les termes importants d'un document donné sont les mots dont les fréquences diffèrent des fréquences suggérées par le modèle d'indépendance.

Dans Lucène, cette approche prend une de ces mesures d'indépendance comme paramètre [30] [33] :

1- La mesure saturée :

$$D_1 = \frac{tf_{ij} - e_{ij}}{e_{ij}}$$

2- la standardisation :

$$D_2 = \frac{(tf_{ij} - e_{ij})^2}{e_{ij}}$$

3- la distance chi-carré normalisée :

$$D_3 = \frac{tf_{ij} - e_{ij}}{\sqrt{e_{ij}}}$$

Tel que : tf_{ij} est la fréquence du terme i dans le document j ;

et $e_{ij} = \frac{TFi \times Dj}{N}$;

Où : TF_i est la fréquence de terme i dans la collection ($TF_i = \sum tf_{ij}$) ;

N est la taille de la collection ;

Dj est la longueur de document j.

3.3.6 IBSimilarity (information-based model) :

Plusieurs travaux en RI se sont attachés à la notion d'information apportée par un terme dans un document. En particulier, plusieurs auteurs, dont l'un des premiers futs Harter (Harter, 1975), ont noté le fait que plus le comportement d'un mot au sein d'un document s'écarte du comportement moyen de ce même mot dans la collection, plus ce mot est significatif pour le document. C'est cette idée, qui est une des idées à la base des modèles DFR, que nous retenons ici pour qualifier les modèles fondés sur l'information comme IB (information based model) qui se calcule de la manière suivante [34] :

$$\mathrm{RSV}(q,d) = \sum_{w \in d \cap q} -x_w^q \log \mathrm{Proba}\,(X \geq t_w^d \mid \lambda)$$

Où : x_w^q Fréquence du terme w dans la requête q ;

x_w^d Fréquence du terme w dans le document d ;

X : est une variable aléatoire qui compte l'occurrence de terme w ;

t_w^d : Version normalisée de x_w^d ;

λ : Paramètre de la distribution de probabilité, qui lors de l'implémentation peut prendre l'une de ces deux valeurs :

- LambdaDFF : nombre moyen de documents où le terme w apparaît.
- LambdaTTF : nombre moyen d'occurrences de terme w dans la collection.

En effet, Lucène offre la possibilité d'établir cette similarité avec ou sans normalisation grâce au paramètre 'Normalization'. Pour tirer le paramètre le plus efficace, nous avons choisis de comparer les résultats de ce modèle avec et sans normalisation, sachant que le modèle de normalisation utilisé suppose une distribution uniforme du terme fréquence. Dans le chapitre suivant, la IBSimilarity avec normalisation sera utilisée sous le nom de 'IB normalisé', tandis que 'IB non normalisé' réfère à IBSimilarity sans normalisation.

3.3.7 LMJelinekMercer Similarity:

C'est un modèle de langue basé sur la méthode de lissage par interpolation Jelinek-Mercer, Cette méthode consiste à combiner un modèle avec un ou des modèles d'ordre inférieur systématiquement, en utilisant un coefficient λ pour contrôler l'influence de chaque modèle, sa valeur optimale dépend à la fois de la collection et de la requête.

La formule de LMJelinek est définie comme suit :

$$RSV(Q,d) = \prod_{t \in Q} (1 - \lambda) P(t \mid M_c) + \lambda\, P(t \mid M_d)$$

Sachant que : λ est un paramètre choisi de telle manière à maximiser l'espérance des données, sa valeur est souvent déterminée avec le processus de maximisation de l'espérance (EM)[2] **[29].**

M_c : modèle générale (collection) ;

M_d : modèle de document ;

Et : $P(t \mid M_c) = P(t) = \frac{Total\_tf}{Total\_tf\_col}$;

Où : Total_tf : fréquence du terme dans la collection ;

Total_tf_col : somme des fréquences de tous les termes de la collection.

3.3.8 Lissage Dirichlet:

Fondé en 2001, ce lissage bayésien est l'une des similarités utilisée par Lucène pour représenter le modèle de langue, il s'appuie sur des distributions de probabilité à priori de Dirichlet, Chengxiang Zhai et John Lafferty en étudiant des méthodes de lissage des modèles de langage appliqués à la recherche d'informations **[30]**.

Dans cette méthode, la fréquence d'un mot m_i dans le document 'D' est incrémentée de $\mu P_{ML}(m_i|D)$, où μ est paramètre appelé pseudo-fréquence **[29]**.

[2] Est un algorithme itératif qui permet de trouver les paramètres du maximum de vraisemblance d'un modèle probabiliste.

La probabilité d'un mot selon le modèle de langue du document devient la suivante :

$$P_{Dir}(m_i|d) = \frac{tf(mi,D) + \mu P_{ML}(m_i|D)}{|D| + \mu}$$

Où |D| est la taille du document (le nombre total d'occurrences de mots) ;

$tf(m_i, D)$ est la fréquence du mot m_i dans le document D.

Cette formule attribue un score négatif aux documents qui contenant le terme, mais avec moins d'occurrences que prévu par le modèle de langage de collection. L'implémentation Lucène revient 0 pour ces documents **[30]**.

3.3.9 L'approche axiomatique :

Basée sur la logique mathématique, la sémantique axiomatique est un mode d'exposition des sciences exactes fondé sur des propositions admises sans démonstration et nettement formulées (appelées axiomes) [35]. Ces derniers vont servir par la suite pour constituer le point de départ de la théorie que l'on propose d'édifier.

L'exploration des approches axiomatiques a était proposé récemment dans le domaine de la recherche d'information par Hui Fang et Chengxiang Zhai **[36]** pour développer des modèles de recherche basés sur la modélisation directe de la pertinence avec un ensemble de propriétés exprimées mathématiquement sous forme de contraintes formelles pour guider la recherche d'une solution optimale de recherche formalisées définies au niveau des termes . Son principe consiste à rechercher dans un espace de fonctions de recherche candidates, un modèle qui peut satisfaire un ensemble de contraintes de recherche raisonnables. Cette approche est implémentée sur Lucène sous forme d'une famille de modèles basés sur BM25, la normalisation de la longueur du document et le modèle de langage avec Dirichlet **[30].** En effet, des paramètres tels que la fréquence des termes et la fréquence inversée du document dans les modèles d'origine sont modifiés de manière à suivre certaines contraintes axiomatiques.

3.4 Conclusion :

Dans ce chapitre, nous avons présenté les modèles principaux de la recherche d'information. D'abord nous avons commencé par introduire les notions de bases de chaque modèle, notamment leurs mécanismes de fonctionnement. Quant aux approches implémentées dans Lucène, nous avons définie l'ensemble des similarités qui seront exploitées dans notre étude.

Le chapitre suivant sera consacré pour l'évaluation et la comparaison de ces similarités pour mettre en évidence les modèles les plus performants.

Chapitre 4

Implémentation et Evaluation.

4.1 Introduction :

Dans ce chapitre, nous présentons les différents outils utilisés dans notre travail ainsi que les deux collections AP89 et WSJ. Nous exposerons également les résultats d'évaluation de notre étude comparative entre les différentes approches de recherche implémentées dans Lucène.

4.2 Outils de développement :

Pour réaliser notre étude comparative, nous avons commencé par étendre certaines classes java de la plateforme Lucène pour indexer les deux corpus AP89 et WSJ. Ensuite nous avons réalisé une comparaison entre les différentes similarités implémentées sur Lucène en s'appuyant sur les résultats d'indexation. Apres cela, nous avons évalué les résultats obtenus sous trec_eval.

4.2.1 L'IDE Eclipse:

Eclipse est une plateforme de développement libre écrite principalement en java, elle favorise le développement des logiciels open sources. Il est connu grâce à son fameux environnement intégré Java, de plus ses plugins permettent de couvrir plusieurs d'autres Langages de programmation comme le PHP et le C/C++.

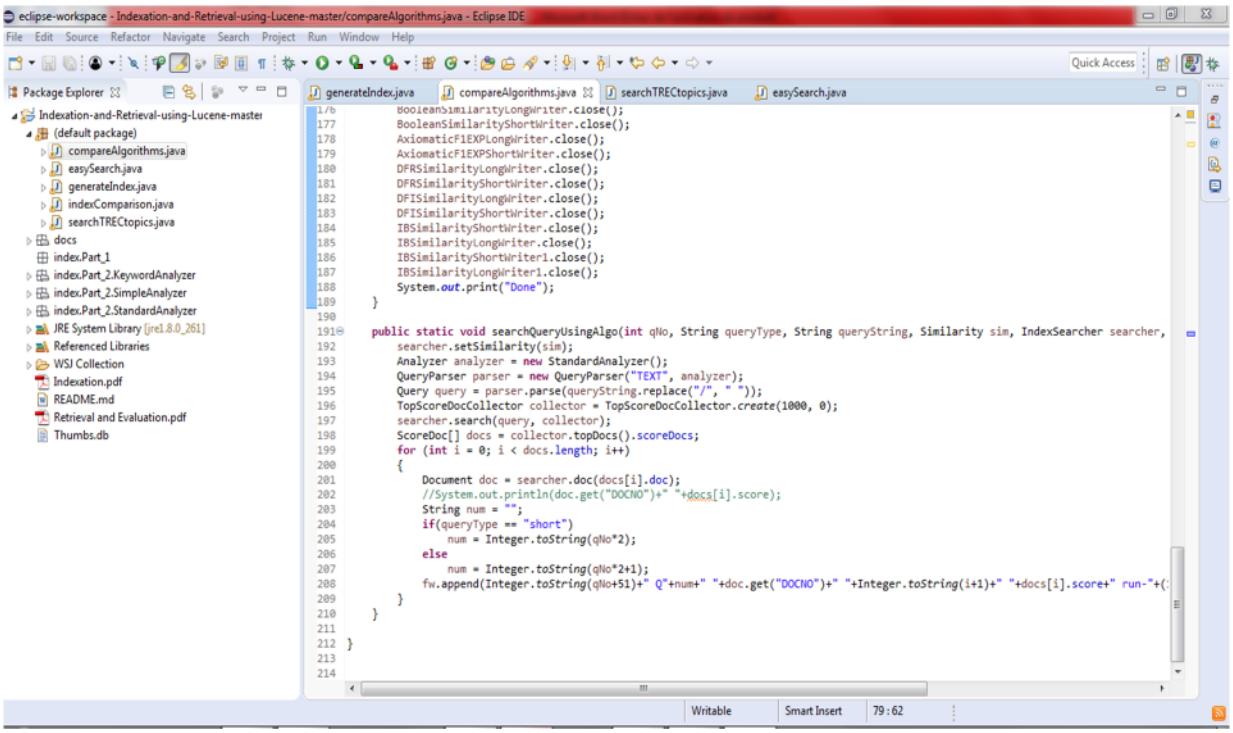

Figure 4.2.1: Interface d'IDE Eclipce

4.2.2 Langage java :

C'est un langage de programmation orienté objet développé par James Gosling et Patrick Naughton, programmeurs de Sun Microsystems. Ce langage permet de développer des applications client-serveur capables de fonctionner sur différents systèmes d'exploitation.

4.2.3 Lucène 8.6.3 :

Lucène est un moteur de recherche textuelle développé entièrement en Java. Contrairement aux autres moteurs de recherches, Lucène n'est pas une application complète, mais plutôt une bibliothèque de codes qui peuvent facilement être utilisées pour ajouter des capacités de recherche aux applications.

4.2.4 Virtual box :

C'est un logiciel de virtualisation de systèmes d'exploitation publié par Oracle. En utilisant le système hôte de l'ordinateur, ce logiciel permet la création d'un ou de plusieurs machines virtuelles dans lesquels s'installent d'autres systèmes invités.Pour effectuer notre évaluation, nous avons utilisé Ubunto 20.04 comme système invité sur la virtualBox.

4.2.5 Ubunto 20.04 :

C'est un système d'exploitation open source basé sur la distribution Linux. Il peut être utilisé sur les ordinateurs et les serveurs.

4.2.6 Trec eval :

Trec_eval est un outil utilisé pour évaluer les classements de document triés par pertinence. L'évaluation repose sur deux fichiers :

1. **Qrels :** contient les jugements de pertinence pour chaque requête ;
2. **Results :** contient le classement des documents renvoyés par le SRI.

Trec_eval est téléchargeable gratuitement à partir de :"http ://trec.nist.gov/trec_eval/".

- **Utilisation :**

Pour pouvoir exécuter Trec_eval, il suffit de taper la commande suivante dans l'invite de commande :

$ ./trec_eval [-q] [-m measure] qrel_file results_file

Où :

- **trec_eval :** est le nom du programme exécutable ;
- **-q :** c'est pour pouvoir matcher les qrels et les résultats ;
- **-m** : affiche uniquement une mesure spécifique ("-m all_trec" affiche toutes les mesures," -m official" est le paramètre par défaut qui affiche uniquement les mesures principales) ;
- **results_file :** chemin du fichier contenant le classement de documents pour chaque requête . il a le format suivant :

query-id Q0 document-id rank score STANDARD

Avec :

- **query _id :** est une séquence alphanumérique pour identifier la requête ;
- **document-id** : est une séquence alphanumérique pour identifier le document jugé ;
- **rank :** est une valeur entière qui représente la position du document dans le classement ;
- **score :** le degré de pertinence entre le document et la requête.

- **qrel_file :** est le chemin du fichier où on trouve les jugements de pertinence pour chaque requête.Il a le format suivant :

query _ID 0 document_ID relevance

Avec :

- **query _ID:** est une séquence alphanumérique pour identifier la requête.
- **document_ID:** est une séquence alphanumérique pour identifier le document jugé.
- **relevance:** est un nombre pour indiquer le degré de pertinence entre le document et la requête (1 pour pertinent et 0 pour non pertinent).

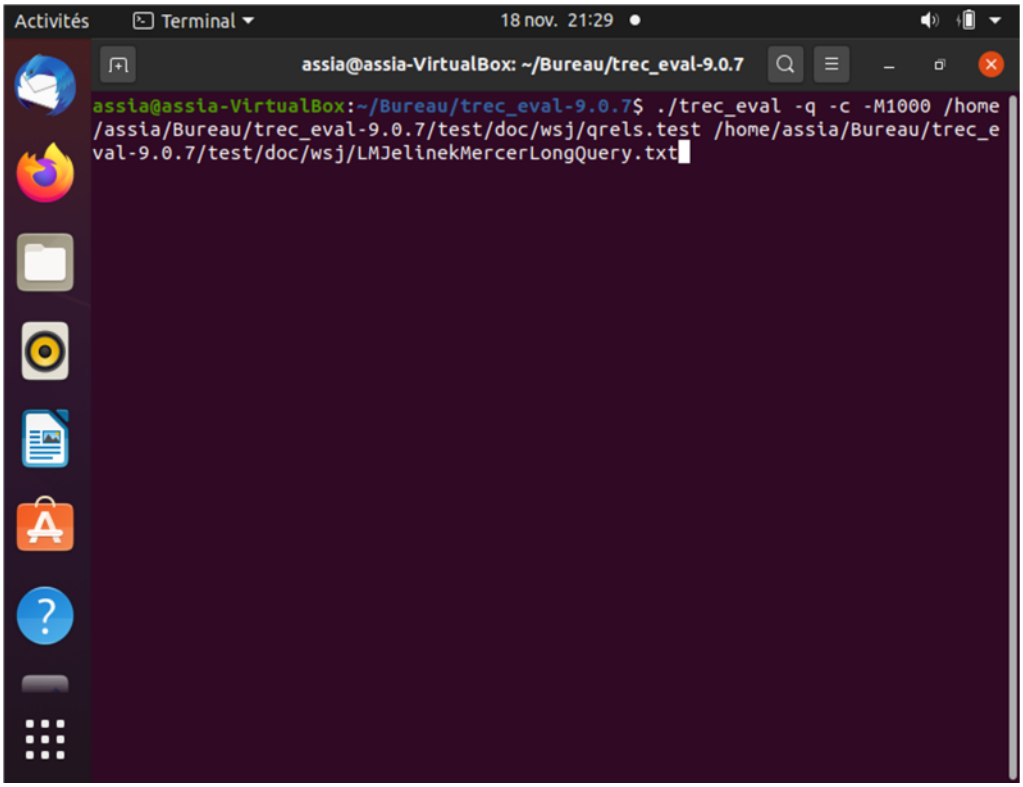

Figure 4.2.2: Exemple de Commande d'exécution de trec_eval.

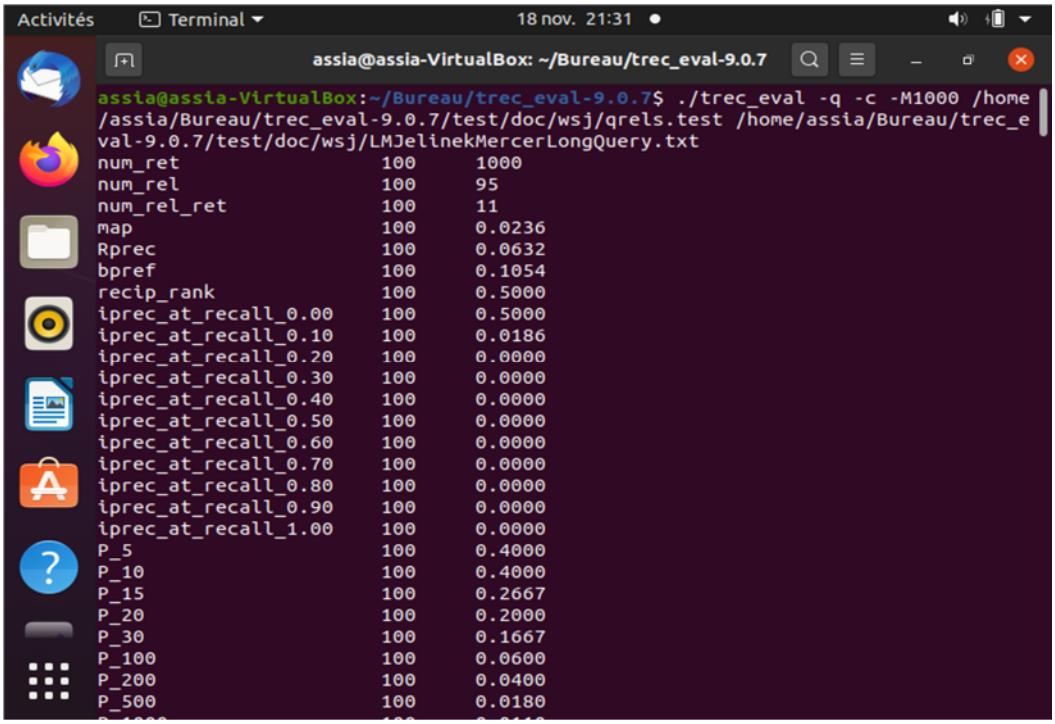

Figure 4.2.1: Exemple Résultat de l'exécution de la commande.

4.3 Présentation des collections :

Afin de réaliser notre étude comparative, nous avons utilisé deux collections de test TREC qui sont AP89 et WSJ leurs données ont été converties au format SGML standard (le format standard des corpus TREC) structuré principalement par des balises comme on le voit dans l'exemple suivant :

```
<DOC>
    <DOCNO> AP891231-0001 </DOCNO>
      <TEXT>
        La collection AP89
      </TEXT>
</DOC>
```

4.3.1 La collection AP89 :

Cette collection de tests est constituée de documents de la presse AP (Associate Press) qui datent de 1989. Elle a été incluse dans les disques TREC 1 [37]. Cette collection est constituée de :

- 84.640 documents.
- 49 requêtes.
- Des jugements de pertinence associés à ces requêtes.

4.3.2 La collection WSJ :

WSJ Wall Street journal (1990, 1991, 1992) est une sous collection de tests constituant des documents qui proviennent de la collection disque TREC 1 [37]**.** Cette sous collection est constituée de :

- 74.520 documents.
- 49 requêtes.
- Des jugements de pertinence associés à ces requêtes.

Tout comme les documents, les requêtes se présentent également sous un format SGML, elles sont structurées par des balises qui nous permettent de distinguer les courtes et les longues requêtes. En effet, les courtes requêtes contiennent des mots clés décrivant le sujet, tandis que les longues requêtes sont des descriptions du sujet qui contiennent également ses mots clés. L'exemple suivant présente respectivement les courtes et les longues requêtes :

<titre> Topic: Airbus Subsidies

<Disc> Description:

Document will discuss government assistance to Airbus Industrie, or mention a trade dispute between Airbus and a U.S. aircraft producer over the issue of subsidies.

Dans ce qui suit, nous présentons les différentes mesures utilisées sur trec eval, avec leur signification.

Table 1 Signification des métriques d'évaluation

Métrique d'évaluation	Signification
P@5	Précision des 5 premiers documents
P@10	Précision des 10 premiers documents
P@20	Précision des 20 premiers documents
P@30	Précision des 30 premiers documents
P@100	Précision des 100 premiers documents
P@500	Précision des 500 premiers documents
P@1000	Précision des 1000 premiers documents
iprec_at_recall 0.00	Moyennes de précision à 0,00 rappel
iprec_at_recall 0.10	Moyennes de précision à 0,10 rappel
iprec_at_recall 0.20	Moyennes de précision à 0,20 rappel
iprec_at_recall 0.30	Moyennes de précision à 0,30 rappel
iprec_at_recall 0.40	Moyennes de précision à 0,40 rappel
iprec_at_recall 0.50	Moyennes de précision à 0,50 rappel
iprec_at_recall 0.60	Moyennes de précision à 0,60 rappel
iprec_at_recall 0.70	Moyennes de précision à 0,70 rappel
iprec_at_recall 0.80	Moyennes de précision à 0,80 rappel
iprec_at_recall 0.90	Moyennes de précision à 0,90 rappel
iprec_at_recall 1	Moyennes de précision à 1 rappel
MAP	La moyenne des précisions moyenne
GMAP	La moyenne géométrique des précisions moyenne
MRR	La moyenne des rangs réciproques
BPref	Préférence Binaire

On précise que rappel à 0.10 par exemple, signifie la récupération de 10% des documents relatif à une requête.

4.4 Résultats d'évaluation de la collection AP89 :

Dans un premier temps, nous allons commencer par présenter les résultats d'évaluation obtenus avec la collection AP89 suivants les 10 similarités utilisées définies dans le troisième chapitre .Ensuite nous allons exploiter ces résultats pour extraire les meilleurs algorithmes.

Les deux premiers tableaux présentent les résultats d'évaluation en utilisant de longues requêtes.

Table 2 Résultats d'évaluation d'AP89 avec de longues requêtes

Métrique d'évaluation	Booléen	Vectoriel	BM25	Dirichlet	Jelinek Mercer
P@5	0.1360	0.2480	0.2640	0.2520	0.2520
P@10	0.1280	0.2280	0.2420	0.2460	0.2320
P@20	0.0970	0.2100	0.2340	0.2320	0.2090
P@30	0.0940	0.1853	0.2207	0.2073	0.1947
P@100	0.0706	0.1286	0.1520	0.1434	0.1374
P@500	0.0494	0.0494	0.0560	0.0554	0.0522
P@1000	0.0199	0.0304	0.0333	0.0325	0.0312
iprec_at_recall 0.00	0.2670	0.4455	0.4563	0.4069	0.4057
iprec_at_recall 0.10	0.1167	0.3110	0.3397	0.3047	0.3203
iprec_at_recall 0.20	0.0839	0.2294	0.2657	0.2632	0.2449
iprec_at_recall 0.30	0.0606	0.1785	0.2341	0.2249	0.2010
iprec_at_recall 0.40	0.0463	0.1527	0.1906	0.1761	0.1687
iprec_at_recall 0.50	0.0358	0.1278	0.1557	0.1504	0.1370
iprec_at_recall 0.60	0.0160	0.0993	0.1330	0.1254	0.1090
iprec_at_recall 0.70	0.0119	0.0636	0.0943	0.0892	0.0831
iprec_at_recall 0.80	0.0092	0.0334	0.0662	0.0629	0.0586
iprec_at_recall 0.90	0.006	0.0176	0.0345	0.0373	0.0350
iprec_at_recall 1	0.000	0.0026	0.0077	0.0130	0.0098
MAP	0.0470	0.1363	0.1655	0.1566	0.1473
GMAP	0.0061	0.0241	0.0337	0.0280	0.0272
MRR	0.2302	0.4281	0.4324	0.3693	0.3696
BPref	0.1940	0.2490	0.2801	0.2769	0.2703

Table 3 Suite des résultats d'évaluation d'AP89 avec de longues requêtes.

Métrique d'évaluation	Axiomatique	DFR	DFI	IB Normalisé	IB NonNormalisé
P@5	0.2400	0.2480	0.2400	0.2600	0.2160
P@10	0.2240	0.2240	0.2160	0.2180	0.1940
P@20	0.1970	0.2050	0.1910	0.1950	0.1880
P@30	0.1873	0.1913	0.1780	0.1747	0.1693
P@100	0.1320	0.1344	0.1264	0.1276	0.1094
P@500	0.0525	0.0510	0.0470	0.0506	0.0436
P@1000	0.0314	0.0305	0.0293	0.0302	0.0267
iprec_at_recall 0.00	0.4505	0.4313	0.4073	0.4137	0.3957
iprec_at_recall 0.10	0.3199	0.3044	0.2892	0.3015	0.2286
iprec_at_recall 0.20	0.2403	0.2306	0.2079	0.2234	0.1933
iprec_at_recall 0.30	0.1900	0.1969	0.1800	0.1801	0.1596
iprec_at_recall 0.40	0.1667	0.1591	0.1377	0.1493	0.1285
iprec_at_recall 0.50	0.1435	0.1345	0.1176	0.1305	0.1059
iprec_at_recall 0.60	0.1165	0.1074	0.0864	0.1022	0.0703
iprec_at_recall 0.70	0.0894	0.0792	0.0646	0.0768	0.0537
iprec_at_recall 0.80	0.0639	0.0485	0.0355	0.0504	0.0331
iprec_at_recall 0.90	0.0425	0.0229	0.0220	0.0245	0.0164
iprec_at_recall 1	0.0141	0.0051	0.0048	0.0028	0.0019
MAP	0.1500	0.1406	0.1271	0.1361	0.1118
GMAP	0.0289	0.0257	0.0186	0.0234	0.0170
MRR	0.4284	0.4154	0.3850	0.3898	0.3710
BPref	0.2741	0.2638	0.2620	0.2589	0.2409

Les deux tableaux ci-dessous présentent les résultats d'évaluation des courtes requêtes pour la collection AP89.

Table 4 Résultats d'évaluation de AP89 avec de courtes requêtes.

Métrique d'évaluation	Booléen	Vectoriel	BM25	Dirichlet	Jelinek Mercer
P@5	0.2040	0.2360	0.3120	0.3400	0.2840
P@10	0.1800	0.2300	0.2960	0.3380	0.2740
P@20	0.1700	0.2180	0.2720	0.2890	0.2540
P@30	0.1467	0.2100	0.2527	0.2680	0.2413
P@100	0.1214	0. 1410	0.1714	0.1708	0.1636
P@500	0.0527	0. 0571	0.0647	0.0642	0.0626
P@1000	0.0296	0. 0348	0.0377	0.0382	0.0366
iprec_at_recall 0.00	0.4444	0.4407	0.5010	0.5352	0.4987
iprec_at_recall 0.10	0.2763	0.3199	0.4016	0.3989	0.3676
iprec_at_recall 0.20	0.2032	0.2481	0.3169	0.3299	0.3040
iprec_at_recall 0.30	0.1618	0.2117	0.2686	0.2709	0.2616
iprec_at_recall 0.40	0.1464	0.1743	0.2277	0.2326	0.2223
iprec_at_recall 0.50	0.1311	0.1523	0.2042	0.2146	0.2017
iprec_at_recall 0.60	0.1139	0.1225	0.1687	0.1752	0.1570
iprec_at_recall 0.70	0.0770	0.0983	0.1397	0.1356	0.1291
iprec_at_recall 0.80	0.0376	0.0731	0.0998	0.0950	0.0918
iprec_at_recall 0.90	0.0250	0.0458	0.0572	0.0591	0.0555
iprec_at_recall 0.10	0.0155	0.0175	0.0204	0.0235	0.0197
MAP	0.1258	0.1583	0.2056	0.2083	0.1960
GMAP	0.0214	0.0299	0.0413	0.0416	0.0365
MRR	0.3695	0.4021	0.4595	0.4739	0.4550
BPref	0.2486	0.2737	0.2992	0.3044	0.2964

Table 5 Suite des résultats d'évaluation de AP89 avec de courtes requêtes.

Métrique d'évaluation	Axiomatique	DFR	DFI	IB Normalisé	IB NON Normalisé
P@5	0.2960	0.2920	0.3400	0.2960	0.3240
P@10	0.2580	0.2680	0.3220	0.2820	0.3220
P@20	0.2540	0.2520	0.2800	0.2570	0.2960
P@30	0.2427	0.2320	0.2607	0.2440	0.2647
P@100	0.1624	0.1640	0.1660	0.1654	0.1734
P@500	0.0623	0.0624	0.0632	0.0629	0.0629
P@1000	0.0372	0.0367	0.0368	0.0372	0.0386
iprec_at_recall 0.00	0.5044	0.4999	0.5223	0.5023	0.5361
iprec_at_recall 0.10	0.3903	0.3803	0.4029	0.3782	0.3907
iprec_at_recall 0.20	0.3106	0.2962	0.3110	0.3331	0.3173
iprec_at_recall 0.30	0.2565	0.2542	0.2535	0.2654	0.2617
iprec_at_recall 0.40	0.2201	0.2160	0.2156	0.2270	0.2261
iprec_at_recall 0.50	0.1975	0.1907	0.1978	0.2084	0.2107
iprec_at_recall 0.60	0.1627	0.1568	0.1589	0.1720	0.1760
iprec_at_recall 0.70	0.1366	0.1303	0.1260	0.1356	0.1284
iprec_at_recall 0.80	0.1039	0.0992	0.0903	0.0988	0.0956
iprec_at_recall 0.90	0.0632	0.0562	0.0538	0.0599	0.0604
iprec_at_recall 1	0.0209	0.0197	0.0196	0.0205	0.0225
MAP	0.2000	0.1925	0.1989	0.2029	0.2027
GMAP	0.0374	0.0364	0.0414	0.0407	0.0439
MRR	0.4757	0.4583	0.4796	0.4555	0.4948
BPref	0.2959	0.2921	0.2928	0.3015	0.3024

4.5 Interprétation des résultats sur la collection AP89 :

- **Précision :**

Dans le cas de longues requêtes, BM25 affiche les meilleurs résultats pour la précision, mais ses résultats sont très comparables avec ceux de Dirichlet notamment si on prend une bonne quantité de documents. Tandis que pour les courtes requêtes les 3 algorithmes Dirichlet, DFI et IBNonNormalisé présentent de résultats très similaires.

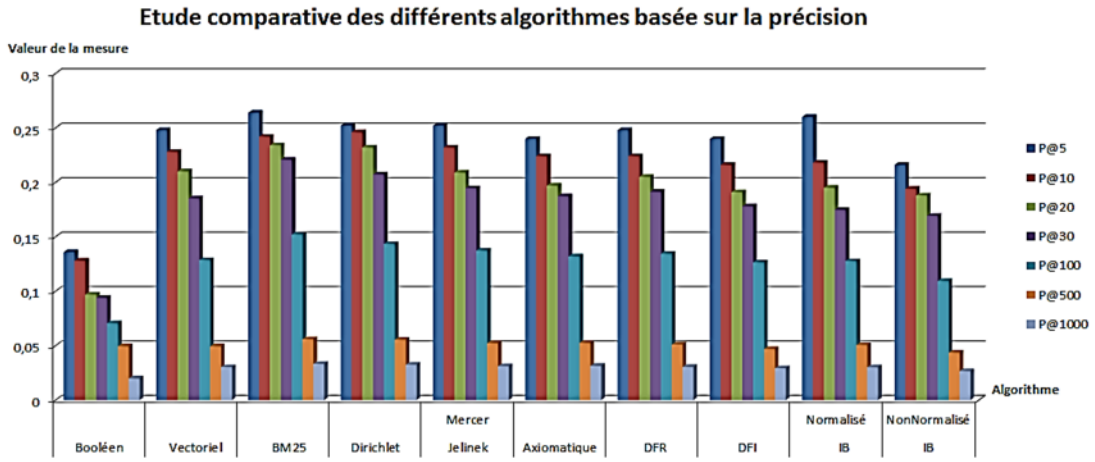

Figure 2.5.1: Histogramme de l'étude comparative de différents algorithmes basée sur la Précision (Long query)

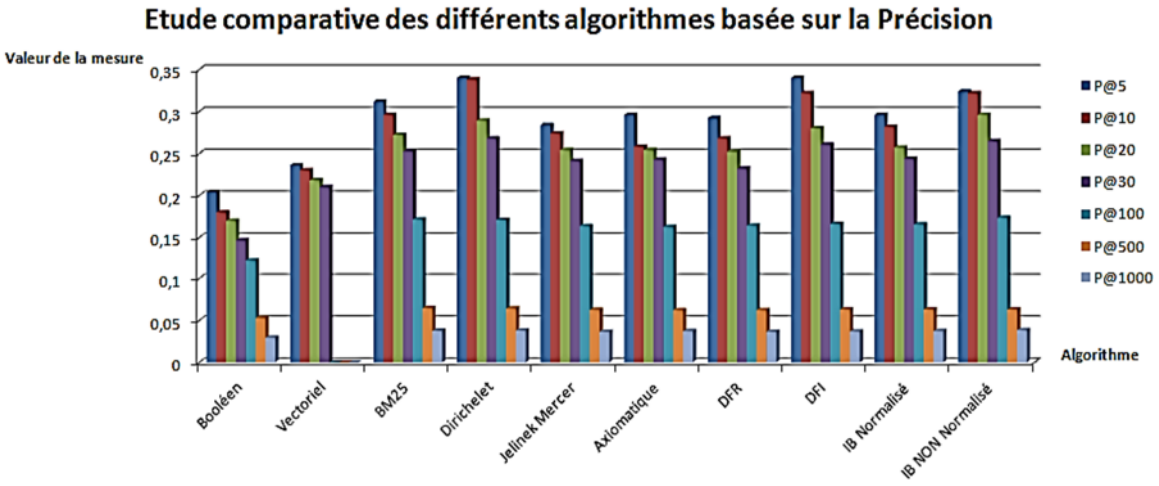

Figure 3.5.2: Histogramme de l'étude comparative de différents algorithmes basée sur la Précision (Short query).

- **Basé sur le Rappel Précision:**

On remarque que BM25 est le meilleur algorithme pour évaluer les longues requêtes, cependant, quand le système retourne plus de 70% des documents relatifs à une requête, ses performances sont très similaires à l'algorithme de Dirichlet et la similarité axiomatique.

Entretemps, Dirichlet affiche les meilleurs résultats pour les courtes requêtes mais ils restent comparables avec ceux de IBnon normalisé. Néanmoins, quand le système retourne 80% de documents relatifs à une requête la similarité axiomatique semble être meilleure.

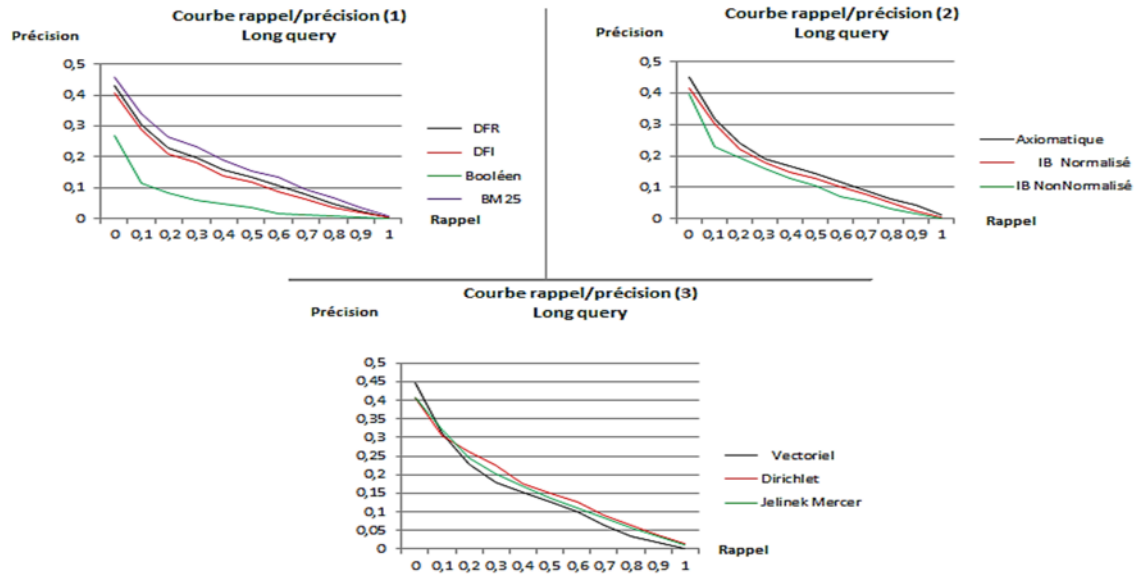

Figure 4.5.3: Courbes Rappel / Précision des différents algorithmes (Long query)

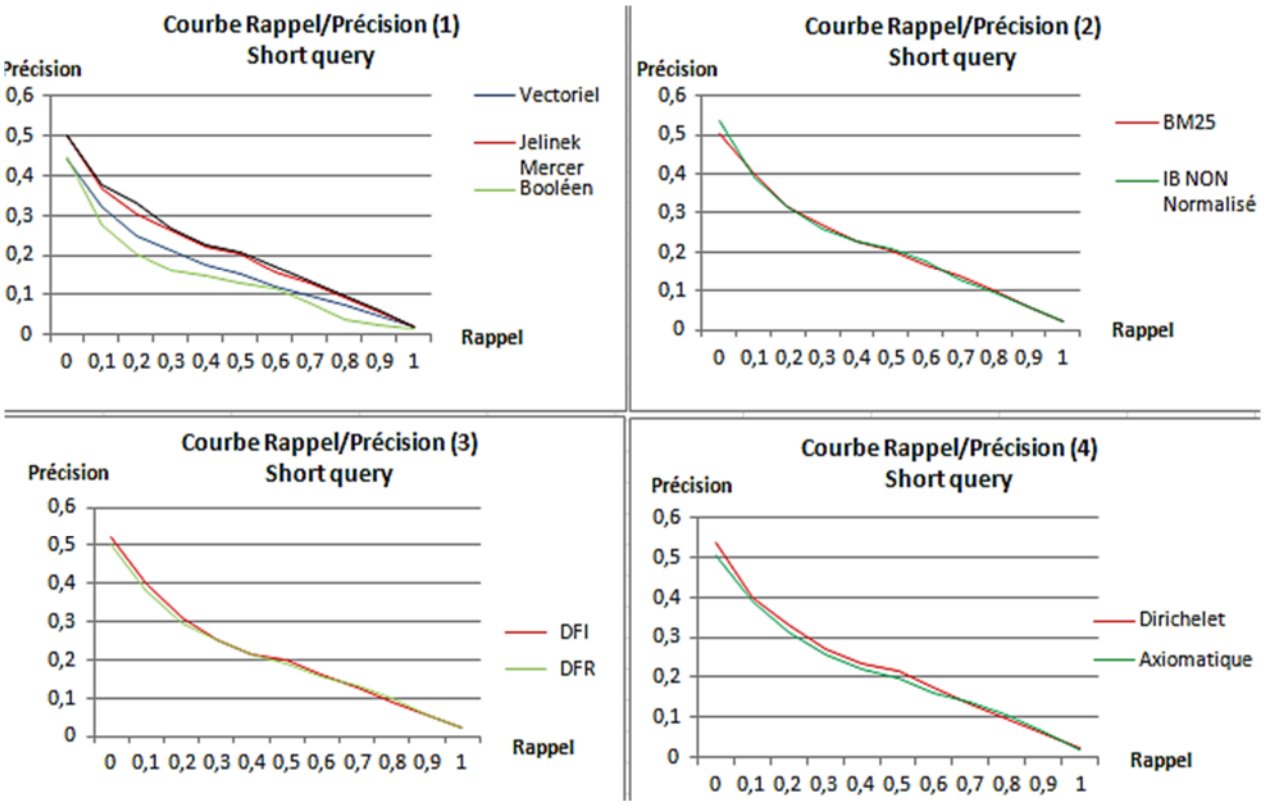

Figure 5.5.4: Courbes Rappel / Précision des différents algorithmes (Short query)

- **MAP :**

En comparant les différentes valeurs de la MAP entre les similarités, BM25 semble être le plus performant pour les longues requêtes. D'autre part, Dirichlet donne des résultats meilleurs que BM25 pour les courtes requêtes.

- **GMAP :**

Pour les longues requêtes, on remarque que les meilleurs résultats sont retournés par BM25. Entretemps, IBNonNormalisé affiche des résultats comparables mais meilleurs par rapport à ceux de Dirichlet pour les courtes requêtes.

- **MRR:**

BM25 affiche de très bons résultats pour la mesure MRR, d'autre part, la similarité axiomatique affiche également de bons résultats. Tandis que pour les courtes requêtes, IBNonNormalisé semble être le meilleur par rapport aux autres algorithmes.

- **BPref :**

Si on prend de longues requêtes, BM25 est le plus performant, mais il reste comparable avec la similarité axiomatique. Concernant les courtes requêtes, Dirichlet et IBNonNormalisé donnent les meilleurs résultats.

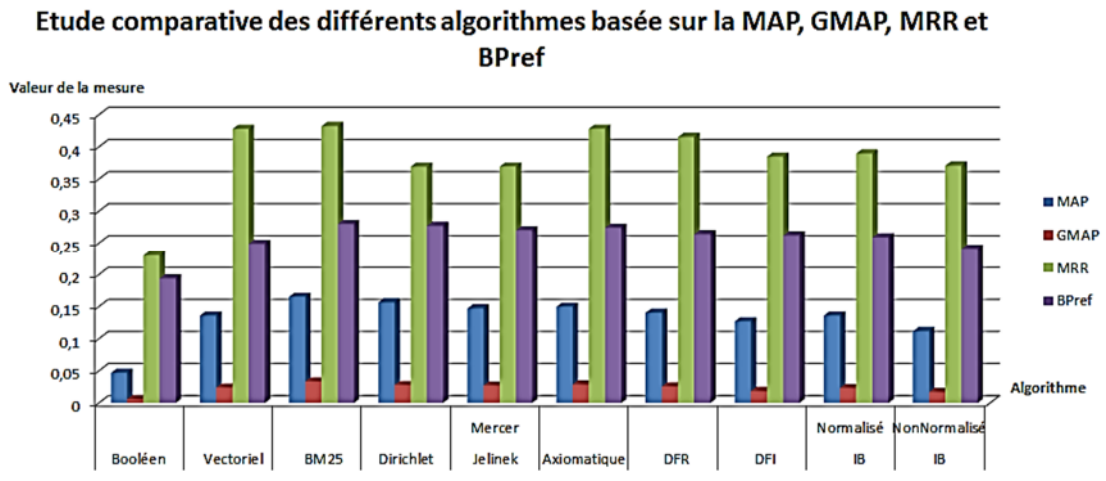

Figure 6.5.5: Histogramme de l'étude comparative de différents algorithmes basée sur la MAP, GMAP, MRR et BPref (Long query).

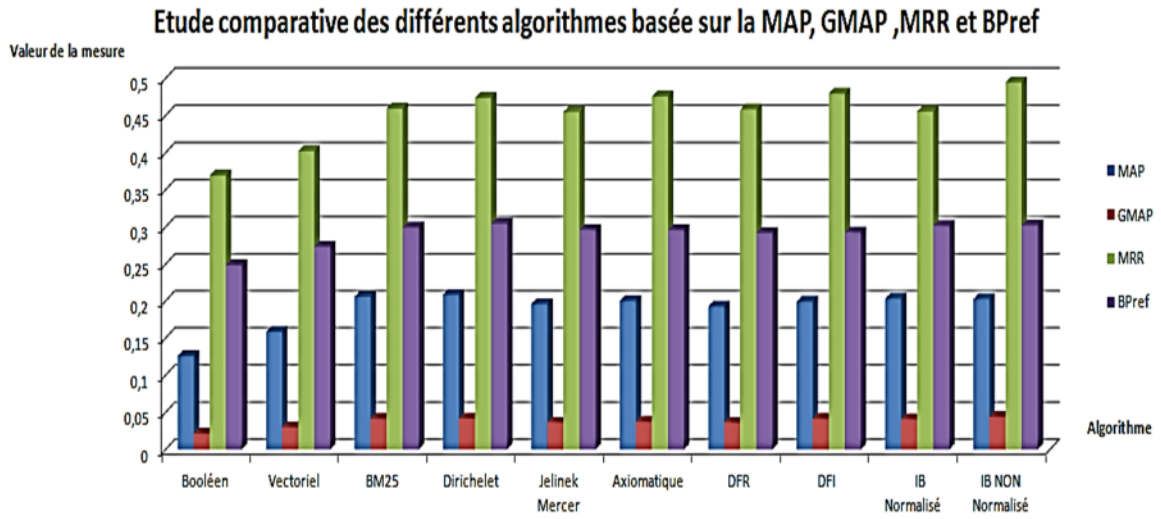

Figure 7.5.6: Histogramme de l'étude comparative de différents algorithmes basée sur la MAP, GMAP, MRR et BPref (Short query).

4.6 Résultats d'évaluation de la collection WSJ :

Pour commencer, nous allons présenter les résultats d'évaluation obtenus avec la collection WSJ suivants les 10 similarités utilisées. Par ailleurs, nous déduirons les meilleurs algorithmes à partir de ces résultats.

Les deux premiers tableaux présentent les résultats d'évaluation en utilisant de longues requêtes.

Table 6 Résultats d'évaluation de WSJ avec de longues requêtes.

Métrique d'évaluation	Booléen	Vectoriel	BM25	Dirichlet	Jelinek Mercer
P@5	0.1200	0.1680	0.2760	0.2480	0.2360
P@10	0.1160	0.1500	0.2540	0.2300	0.2360
P@20	0.1010	0.1430	0.2190	0.1840	0.1890
P@30	0.0867	0.1353	0.1927	0.1667	0.1740
P@100	0.0572	0.0852	0.1132	0.0962	0.1020
P@500	0.0224	0.0334	0.0392	0.0348	0.0356
P@1000	0.0143	0.0208	0.0230	0.0218	0.0221
iprec_at_recall 0.00	0.2484	0.3138	0.4778	0.4678	0.4338
iprec_at_recall 0.10	0.0652	0.1245	0.2311	0.1939	0.1973
iprec_at_recall 0.20	0.0270	0.0547	0.1293	0.1093	0.1050
iprec_at_recall 0.30	0.0111	0.0269	0.0881	0.0600	0.0592
iprec_at_recall 0.40	0.0011	0.0053	0.0437	0.0234	0.0245
iprec_at_recall 0.50	0.0004	0.0023	0.0188	0.0088	0.0182
iprec_at_recall 0.60	0.0003	0.0004	0.0100	0.0056	0.0032
iprec_at_recall 0.70	0.0003	0.0000	0.0100	0.0038	0.0022
iprec_at_recall 0.80	0.0000	0.0000	0.0000	0.0006	0.0000
iprec_at_recall 0.90	0.0000	0.0000	0.0000	0.0000	0.0000
iprec_at_recall 1	0.0000	0.0000	0.0000	0.0000	0.0000
MAP	0.0165	0.0320	0.0682	0.0539	0.0526
GMAP	0.0023	0.0069	0.0122	0.0114	0.0117
MRR	0.2371	0.2858	0.4439	0.4379	0.3824
BPref	0.0703	0.1175	0.1318	0.1210	0.1271

Table 7 Suite des résultats d'évaluation de WSJ avec de longues requêtes.

Métrique d'évaluation	Axiomatique	DFR	DFI	IB Normalisé	IB NonNormalisé
P@5	0.2600	0.2280	0.1920	0.2400	0.1520
P@10	0.2240	0.2220	0.1940	0.2180	0.1500
P@20	0.1970	0.1950	0.1760	0.1920	0.1290
P@30	0.1693	0.1700	0.1600	0.1733	0.1140
P@100	0.0970	0.0976	0.0954	0.1024	0.0712
P@500	0.0338	0.0343	0.0325	0.0350	0.0264
P@1000	0.0210	0.0209	0.0206	0.0212	0.0163
iprec_at_recall 0.00	0.4397	0.4296	0.3856	0.4429	0.2798
iprec_at_recall 0.10	0.2217	0.2107	0.1768	0.1766	0.0900
iprec_at_recall 0.20	0.1112	0.0945	0.0776	0.0936	0.0402
iprec_at_recall 0.30	0.0734	0.0548	0.0409	0.0543	0.0205
iprec_at_recall 0.40	0.0396	0.0220	0.0101	0.0198	0.0053
iprec_at_recall 0.50	0.0171	0.0141	0.0055	0.0132	0.0009
iprec_at_recall 0.60	0.0110	0.0009	0.0010	0.0023	0.0006
iprec_at_recall 0.70	0.0073	0.0007	0.0005	0.0010	0.0002
iprec_at_recall 0.80	0.0000	0.0000	0.0000	0.0000	0.0000
iprec_at_recall 0.90	0.0000	0.0000	0.0000	0.0000	0.0000
iprec_at_recall 1	0.0000	0.0000	0.0000	0.0000	0.0000
MAP	0.0607	0.0526	0.0433	0.0497	0.0246
GMAP	0.0107	0.0105	0.0084	0.0104	0.0035
MRR	0.4111	0.3915	0.3549	0.4163	0.2659
BPref	0.1292	0.1209	0.1147	0.1199	0.0770

Les deux tableaux ci-dessous présentent les résultats d'évaluation des courtes requêtes pour la collection WSJ.

Table 8 Résultats d'évaluation de WSJ avec de courtes requêtes.

Métrique d'évaluation	Booléen	Vectoriel	BM25	Dirichlet	Jelinek Mercer
P@5	0.2080	0.1680	0.2920	0.3120	0.2800
P@10	0.1730	0.1500	0.2640	0.2740	0.2380
P@20	0.1620	0.1430	0.2160	0.2330	0.2160
P@30	0.1420	0.1353	0.1900	0.2080	0.1893
P@100	0.0818	0.0852	0.1172	0.1174	0.1136
P@500	0.0321	0.0334	0.0390	0.0392	0.0379
P@1000	0.0202	0.0208	0.0235	0.0230	0.0229
iprec_at_recall 0.00	0.3988	0.3138	0.4450	0.5191	0.4354
iprec_at_recall 0.10	0.1347	0.1245	0.2492	0.2244	0.2255
iprec_at_recall 0.20	0.0817	0.0547	0.1372	0.1225	0.1289
iprec_at_recall 0.30	0.0468	0.0269	0.0836	0.0917	0.0810
iprec_at_recall 0.40	0.0291	0.0053	0.0365	0.0389	0.0381
iprec_at_recall 0.50	0.0114	0.0023	0.0151	0.0160	0.0161
iprec_at_recall 0.60	0.0106	0.004	0.0112	0.0112	0.0112
iprec_at_recall 0.70	0.0105	0.0000	0.0100	0.0100	0.0100
iprec_at_recall 0.80	0.0000	0.0000	0.0000	0.0000	0.0000
iprec_at_recall 0.90	0.0000	0.0000	0.0000	0.0000	0.0000
iprec_at_recall 1	0.0000	0.0000	0.0000	0.0000	0.0000
MAP	0.0451	0.0320	0.0694	0.0694	0.0671
GMAP	0.0081	0.0066	0.0122	0.0128	0.0106
MRR	0.3730	0.2858	0.4051	0.4875	0.3935
BPref	0.1126	0.1175	0.1320	0.1276	0.1311

Table 9 Suite des résultats d'évaluation de WSJ avec de courtes requêtes.

Métrique d'évaluation	Axiomatique	DFR	DFI	IB Normalisé	IB NonNormalisé
P@5	0.2760	0.2680	0.2770	0.3000	0.3040
P@10	0.2540	0.2400	0.2720	0.2460	0.2580
P@20	0.2190	0.2040	0.2350	0.2130	0.2200
P@30	0.1927	0.1820	0.2060	0.1935	0.1967
P@100	0.1132	0.1112	0.1148	0.1158	0.1146
P@500	0.0392	0.0381	0.0385	0.0392	0.0382
P@1000	0.0230	0.0227	0.0235	0.0233	0.0226
iprec_at_recall 0.00	0.4778	0.4345	0.4785	0.4578	0.5082
iprec_at_recall 0.10	0.2311	0.2139	0.2272	0.2343	0.2069
iprec_at_recall 0.20	0.1293	0.1271	0.1235	0.1329	0.1096
iprec_at_recall 0.30	0.0881	0.0762	0.0807	0.0842	0.0815
iprec_at_recall 0.40	0.0437	0.0361	0.0345	0.0388	0.0332
iprec_at_recall 0.50	0.0188	0.0152	0.0132	0.0156	0.0135
iprec_at_recall 0.60	0.0100	0.0112	0.0112	0.0112	0.0103
iprec_at_recall 0.70	0.0100	0.0100	0.0100	0.0100	0.0103
iprec_at_recall 0.80	0.0000	0.0000	0.0000	0.0000	0.0000
iprec_at_recall 0.90	0.0000	0.0000	0.0000	0.0000	0.0000
iprec_at_recall 1	0.0000	0.0000	0.0000	0.0000	0.0000
MAP	0.0682	0.0645	0.0670	0.0689	0.0633
GMAP	0.0122	0.0104	0.0115	0.0116	0.0130
MRR	0.4439	0.3889	0.4443	0.4138	0.4742
BPref	0.1318	0.1281	0.1315	0.1318	0.1218

4.7 Interprétation des résultats sur la collection WSJ :

- **Précision :**

Pour les longues requêtes, BM25 est plus performant que la similarité axiomatique qui donne également de bons résultats. Entretemps, Dirichlet et DFI retournent des meilleurs résultats pour les courtes requêtes.

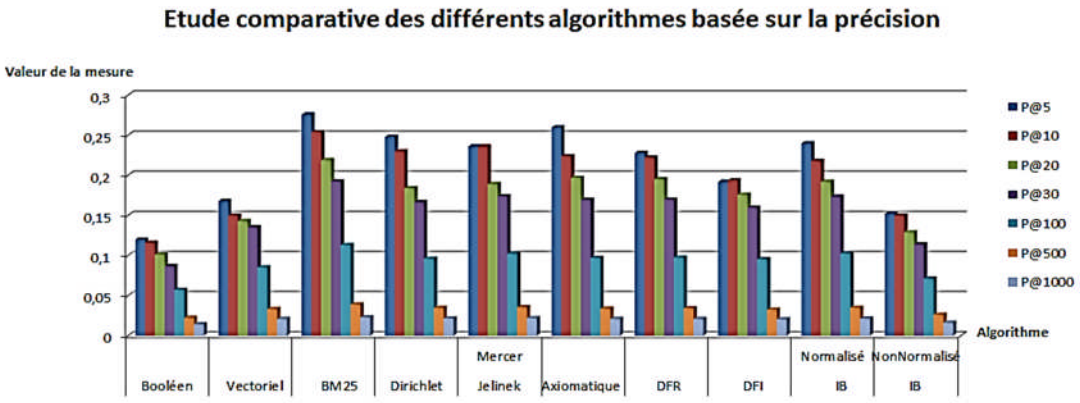

Figure 8.7.1: Histogramme de l'étude comparative de différents algorithmes basée sur la Précision (Long query).

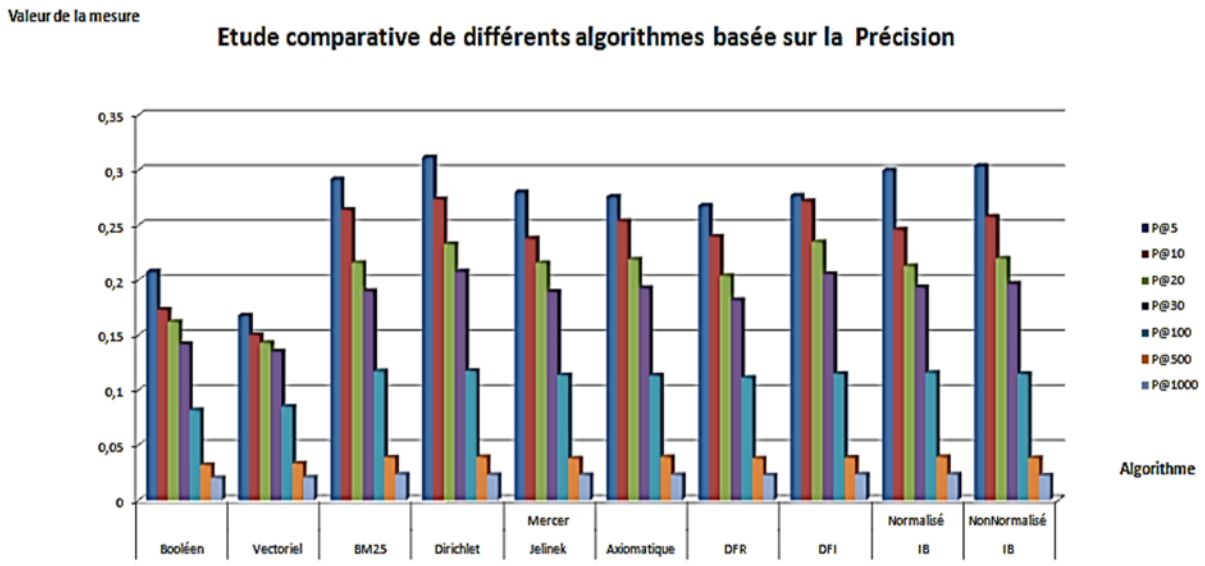

Figure 9.7.2: Histogramme de l'étude comparative de différents algorithmes basée sur la Précision (Short query).

- **Basé sur le Rappel Précision:**

Pour les longues requêtes, on remarque que les meilleurs résultats sont retournés par BM25. Tandis que pour les courtes requêtes, les trois algorithmes Dirichlet, BM25 et Axiomatique sont les plus performants.

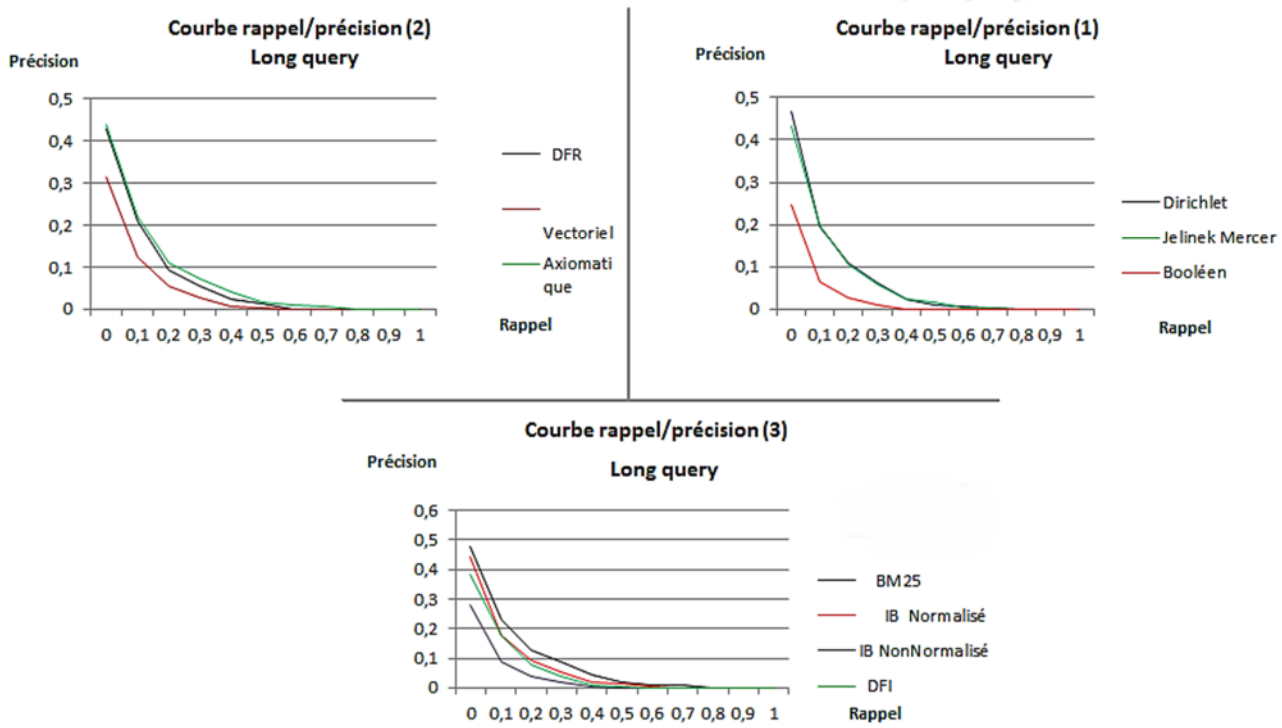

Figure 10.7.3: Courbes Rappel /Précision de différents algorithmes (Long query).

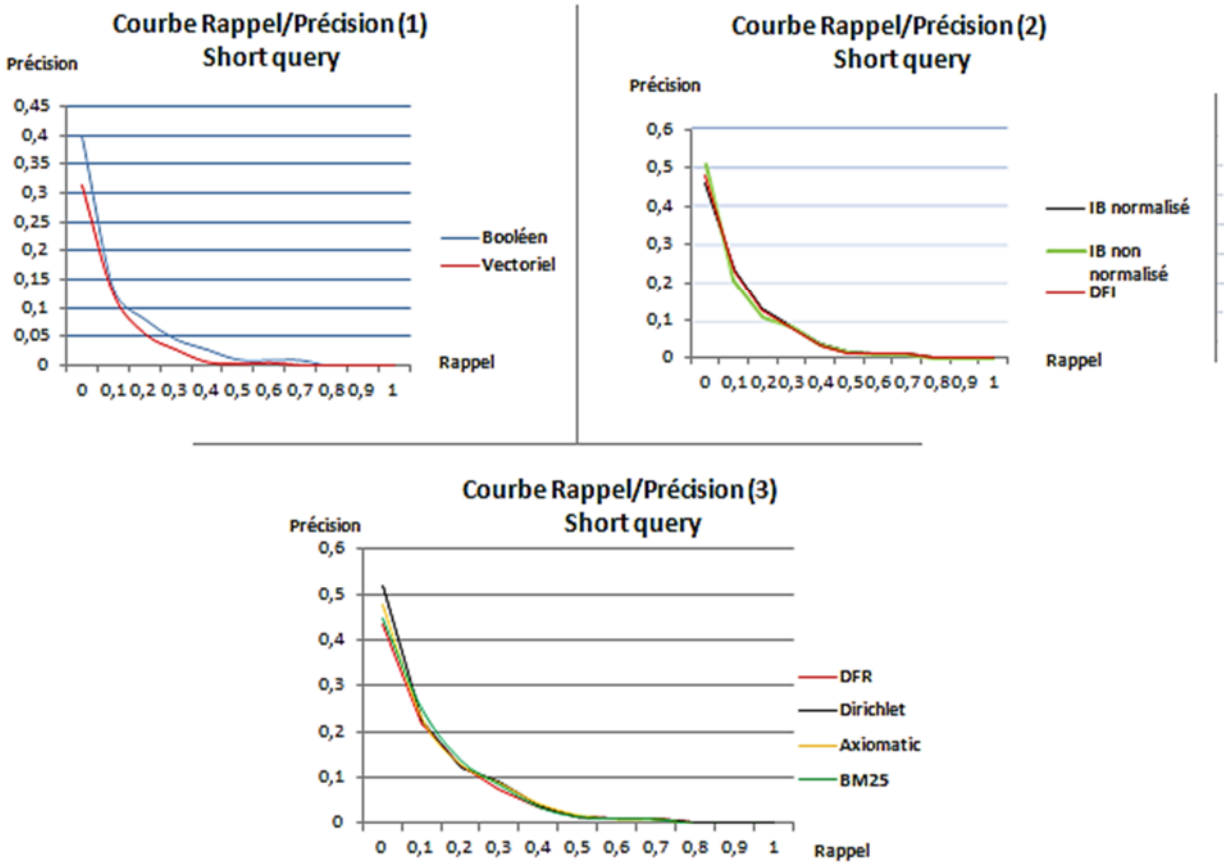

Figure 11.7.4: Courbes Rappel /Précision de différents algorithmes (Short query).

- **MAP:**

Concernant les longues requêtes, BM25 affiche les meilleurs résultats par rapport aux autres algorithmes, Néanmoins ses performances sont similaires à celles de Dirichlet pour les courtes requêtes.

- **GMAP:**

Si on prend de longues requêtes, on remarque que les trois algorithmes BM25, Dirichlet et Jelinek sont plus efficaces. D'autre part Dirichlet affiche les meilleurs résultats pour les courtes requêtes, qui restent très proches de BM25 et la similarité axiomatique.

- **MRR:**

BM25 affiche les meilleurs résultats pour les longues requêtes, entretemps Dirichlet et la similarité axiomatique retournent également de bons résultats. Alors que pour les courtes requêtes, Dirichlet affiche des résultats remarquables qui restent comparables avec ceux de l'algorithme IB non Normalisé.

- **BPref:**

En prenant de longues requêtes, BM25 retourne les meilleurs résultats. Cependant, la similarité axiomatique et IB non normalisé semblent être plus efficaces que DFI qui affiche également des résultats très similaires pour les courtes requêtes.

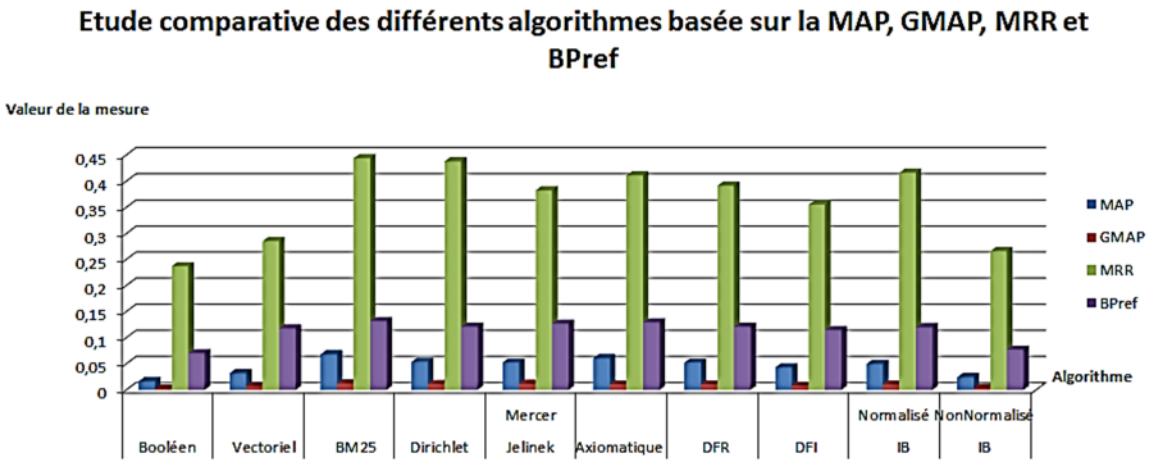

Figure 12.7.5: Histogramme de l'étude comparative de différents algorithmes basée sur la MAP, GMAP, MRR et BPref (Long query).

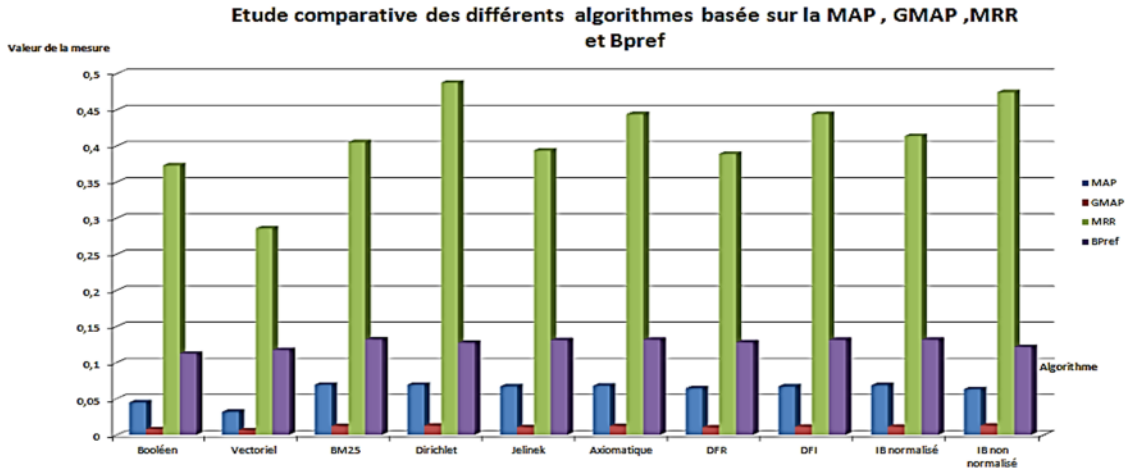

Figure 13.7.6: Histogramme de l'étude comparative de différents algorithmes basée sur la MAP, GMAP, MRR et BPref (Short query).

D’après les résultats d’évaluation des différentes approches sur les deux collections de test AP89 et WSJ, nous constatons que BM25 et Dirichlet sont les similarités les plus performantes. Par ailleurs elles ont pu remédier à plusieurs lacunes des approches classiques en proposant leurs propres disciplines.

Le modèle de Dirichlet tire sa performance des techniques de lissage qui éliminent les probabilités nulles. Alors que BM25 utilise des techniques de normalisation de la longueur des documents, en considérant par exemple le TF logarithmique, au lieu du TF brut. En outre, il est prouvé qu’il fonctionne très bien sur les collections de documents, particulièrement celles conçues par TREC. En raison de ses performances, BM25 est la similarité par défaut de plusieurs moteurs de recherche y compris Lucéne, Elasticsearch et Solr.

Sur la base de ces techniques, BM25 et Dirichlet réalisent souvent de meilleures performances par rapport aux autres approches.

4.8 Conclusion :

Dans ce dernier chapitre, nous avons défini les outils nécessaires pour la réalisation de notre travail. En outre nous avons présenté un aperçu de notre évaluation, et nous avons terminé avec l’interprétation des résultats obtenus.

Conclusion générale :

Dans notre travail nous nous sommes intéressées aux différentes approches de la recherche d'information implémentées dans Lucéne, qui a pour objectif de tirer les modèles les plus performants.

Au premier lieu, nous avons commencé par introduire les notions fondamentales liées à la recherche d'information, ensuite nous avons présenté le moteur de recherche Lucéne ainsi que les approches principales de la recherche d'information.

Enfin, nous avons élaboré une évaluation d'une dizaine de modèles de recherche sur les collections AP89 et WSJ, et nous avons constaté que BM25 et Dirichlet sont les approches de recherche implémentées dans Lucéne, qui présentent les meilleures performances.

Bibliographie

[1] M. J. Salton G, Introduction to Modern Information Retrieval, McGraw-Hill, Inc, 1986.

[2] J. S. M. Boughanem, Hermès-Lavoisier , 2008. [En ligne]. Available: https://www.irit.fr/~Mohand.Boughanem/Enseignements_RI.php.

[3] Hammache, *Recherche d'Information : un modèle de langue combinant mots simples et mots composés,* Tizi ouzou, Université Mouloud Mammeri, 2012.

[4] R. Mihalsea. [En ligne]. Available: https://web.eecs.umich.edu.

[5] N. Craswell, 2009. [En ligne]. Available: https://link.springer.com/referenceworkontry.

[6] NIST, «Common Evalluation Measure,» [En ligne]. Available: https://trec.nist.gov.

[7] SIGIR, «la recherche et le développement dans la recherche d'informations,» chez *Association for Computing Machinery*, New York, 2008.

[8] S. D. D. K. Alain Baccini, «Analyse des critères d'évaluation des systèmes de recherche d'information,» 2010.

[9] F. Apache, «Lucene,» 20 07 2015. [En ligne]. Available: http://www.open-source-guide.com.

[10] I. SARL, «Apache Lucene: recherche libre pour votre site web,» Digital guide, 21 07 2020. [En ligne]. Available: http://www.ionos.fr. [Accès le 30 08 2020].

[11] F. The Apache Software, «Apache Luce Core,» Apache Lucene, 2011. [En ligne]. Available: https://lucene.apache.org. [Accès le 12 09 2020].

[12] W. W. Y. G. Qi Wang, «The Application of Lucene in Information Leakage Monitoring and Querying System,» School of Computer Science and Technology Beijing University of Posts and Telecommunications Beijing, China, 100876, Decembre 2010. [En ligne]. Available: https://www.researchgate.net.

[13] R. Guy, «Crèer un moteur de recherche avec lucene,» Devloppez.com, 12 06 2006. [En ligne]. Available: http://www.gfx.devloppez.com. [Accès le 16 9 2020].

[14] J. S. M.Boughanem, Hermès-Lavoisier, 2008. [En ligne]. Available: https://www.irit.fr/~Mohand.Boughanem/slides/RI/chap4-mod-bool-vect.pdf.

[15] G. w. Salton, Extended Boolean Information Retrieval, Communication of the ACM, 1983.

[16] S. G, Introduction to Modern Information Retrieval, 1983.

[17] W. S.K.M Wong, «Generalized Vector Space,» chez *Model in Information Retrieval.In Proc of the 8th ACM SIGIR Conference on Research and Development*, New York USA, 1985.

[18] K. Kwok, A neural network for probabilistic information, In ACM SIGIR Forum, 1989.

[19] V. Bernard, «Le connexionnisme,» 2 04 2006. [En ligne]. Available: https://halshs.archives-ouvertes.fr.

[20] L. T. Mohand Boughanem, Connexionisme et génétique pour la recherche d'information, Lavoisier: Hermes-Lavoisier, 2004.

[21] D. F. L. Deerwester.S, Indexing by latent semantic analysis, Journal of the American society for information science, 1990.

[22] G. Kamel, «Modèles de Recherche d'information basés sur les Réseaux Bayésiens et les les Réseaux Possibilistes,» Universitè de Sfax Faculté des Sciences Economiques et de gestion, Sfax, 2017.

[23] Boughanem.M, «chapitre 6:Modele probabiliste,» https://www.irit.fr.

[24] K. L. Maron.M. E, «On relevance,probabilistic indexing and information retrieval,» Journal of the ACM (JACM), 1960.

[25] Abdelkarim.Herzallah, «La recherche d'information,» Universitè de Boumerdes.

[26] H. S. Chris Manning, «Foundations of Statistical Natural Language Processing,» MIT Press., Cambridge, May 1999.

[27] Frederick.Jelinek, «Statistical Methods for Speech Recognition,» MIT Press, 1997.

[28] S. A. D. P. J. D. P. R. L. M. Peter F.Brown, «The Mathematics of Statistical Machine Translation: Estimation, Computational Linguistics,» 1993.

[29] W. K.-Y. N. Mohand Boughanem, *Modèles de langue pour la recherche d'information,* Institut de Recherche en Informatique de Toulouse,TNO TPD,Université de Montréal.

[30] Apache.Lucene, «Class Similarity,» [En ligne]. Available: https://lucene.apache.org/core/8_6_3/core/org/apache/lucene/search/similarities/Similarity.html.

[31] V. Claveau, Vectorisation, Okapi et calcul de similarité pour le TAL : pour oublier enfin le TF-IDF, 2012.

[32] Mohand.Boughanem, «Chapitre 9: DFR,» https://www.irit.fr. [En ligne]. Available: https://www.irit.fr.

[33] B. Taner, IRRA at TREC 2012: Divergence From Independence (DFI), Department of Computer Engineering,Department of Statistics,Mu˜gla University.

[34] E. G. Stéphane Clinchant, Modèles de RI fondés sur l'information, Laboratoire d'Informatique de Grenoble, Université de Grenoble.

[35] Georges.GLAESER, «Axiomatique,» [En ligne]. Available: Encyclopædia Universalis [en ligne]. [Accès le 15 11 2020].

[36] SIGIR.ACM, «International Symposium on Information Retrieval,» Canberra,Australia, 2005.

[37] Harman.Donna, OVERVIEW OF TREC-1, National Institute of Standards and Technologie Gathersburg,Md.20899.