République Démocratique de Madagascar
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique
Université de Toamasina

Faculté des Sciences et Technologie (FST)
Département d'Informatique

Thèse de Master

Domaine : Mathématique, Informatique et Applications

Filière : Informatique

Option : Génie Informatique

Présenté par : BOUDI Gislain Carino Rodrigue

Thème

Moteur de recherche basé sur le modèle vectoriel pour améliorer la recherche des thèses malagasy

Soutenue publiquement, le 26/06/2023, devant le jury composé de :

Mme.	Prénom NOM	MC(B)	(Université de Laghouat)	President
Mlle.	Prénom NOM	MC(A)	(Université de Laghouat)	Examinateur
Mr.	Prénom NOM	MC(B)	(Université de Laghouat)	Examinateur
Mr.	Jérome VELO	Professeur(B)	(Université de Toamasina)	Encadreur

Année Universitaire 2022/2023

Résumé

Resumé

Remerciements

Remerciements

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Vue d'ensemble

La recherche d'information (*RI*) est le fait de rechercher une information dans une base de documents (corpus) a travers un système de recherche d'information ou moteur de recherche. C'est aussi trouver un document pertinent qui satisfait un besoin d'information de l'utilisateur. Dans le cadre de ce devoir, on vise a satisfaire le besoin d'information qui est de trouver des travaux de recherche connexe (articles, thèses doctorat, mémoire de master, etc.) a un thème de recherche afin de pouvoir rédiger la cadre théorique ou état de l'art. Il est donc nécessaire de consulter des ressources sur internet par le biais des moteurs de recherches ou des ressources physiques (Documents imprimés).

Un Système de Recherche d'Information (*SRI*) ou moteur de recherche est un ensemble de programmes qui stock les documents et permet de donner des résultats suivant un degré de pertinence en fonction d'une requête de l'utilisateur.

Le modèle vectoriel est un modèle mathématique utilisé dans la recherche d'information pour définir la comportement de la Système de Recherche d'Information et définit l'algorithme de similarité qui détermine la pertinence des documents dans la collection par rapport a la requête de l'utilisateur afin de classer les résultats (liste des documents) retournés a l'utilisateur.

Contexte de la recherche de thèses malagasy

Pour rechercher des thèses malagasy (soutenue dans les universités a Madagascar), les moteurs de recherche académique comme Google scholar, HAL, Theses.fr, etc., ne sont pas véritablement efficace dans le sens que le nombre des documents (Thèses, Articles) malagasy qui y sont indexé n'est pas suffisant. Cette manque peut être dû au fait que ces ressources ne sont pas déposé en ligne pour faciliter l'accès en ligne, alors qu'ils sont difficile a indexer par les moteurs de recherche. Ce qui implique qu'il faudra passer par un SRI local pour héberger ces documents ainsi que de les rechercher.

Actuellement, le premier système en place est « Thèse malgache en ligne » qui représente **31 160** documents provenant des six universités publiques dont **197** pour l'Université d'Antsira-

nana, **1 190** pour l'Université de Mahajanga, **973** pour l'Université de Toamasina, **27 789** pour l'Université d'Antananarivo, **338** pour l'Université de Fianarantsoa et **673** pour l'Université de Toliara (Université d'Antananarivo, 2016).

Ces documents sont pas encore suffisant, vu le nombre d'étudiants qui termine leurs études en master, doctorat, etc. D'autre part, l'université d'Antananarivo occupe environ le *89%* du nombre des documents totale et le *11%* pourcent restant pour les autres universités. D'autre source tel que des bibliothèques en ligne, et hors-lignes (documents papiers, numériques mais qui ne sont pas accessible en ligne) des universités pouvant être utilisés, mais la plupart de ces systèmes (version en ligne) sont sans interface pour faire des recherches par mots clés. Ces lacunes sont la source de motivation de ce mémoire pour apporter une amélioration.

Problématique

Alors l'identification et l'exploitation des thèses et mémoires malagasy dans un domaine spécifique d'un sujet de recherche sont encore un défi en raison de limitations des ces systèmes de recherche existants. Certains de ces systèmes sont souvent peu performants, difficiles à utiliser en raison d'une interface utilisateur peu conviviale, et limités en terme de ressources et limités aux ressources des six universités publiques malgaches (Si nécessaire). Par conséquent, il est difficile de mener une recherche approfondie et d'exploiter pleinement les résultats disponibles. De plus, ces travaux de recherche sont souvent peu visibles sur les systèmes de recherche d'information populaires tels que *Google Scholar*, *Theses.fr*, *HAL*, *Mémoire online*, etc. Cette situation limite l'exploitation et la valorisation des travaux pertinents réalisés par les étudiants et chercheurs malagasy. Et enfin, la sécurité de ces fruits de recherche est un véritable défi pour les SRI comme la protection de droit d'auteur, protection contre le vol de compétence, la sécurité de l'hébergement, etc.

Objectifs de la mémoire

L'objectif principal est de permettre aux chercheurs malagasy d'accéder facilement aux travaux de recherche pertinents et de promouvoir la recherche locale avec une meilleur sécurité. Ce mémoire vise donc a développer un système de recherche d'information (*SRI*), basé sur le modèle vectoriel, permettant de résoudre les problèmes rencontrés dans l'accès aux travaux de recherche malagasy tels que les mémoires et thèses. Ce système doit être convivial, performant et regrouper tous les documents provenant des établissements d'enseignement supérieur publics et privés. Ainsi ce système propose un accès plus sécurisé pour les documents pour éviter le vol et protéger le droits d'auteur par une mise en place de système d'authentification, un système d'accès par une organisation ou université, et un système de paiement pour un document payant. Pour minimiser les risques, ce SRI doit être hébergé a Madagascar pour protéger ces ressources.

Organisation du devoir

Pour mieux élaborer ce thème, ce devoir va se dérouler en quatre partie. La première partie concerne l'état de l'art qui est divisé en quatre chapitre tel que la Recherche d'Information où on va voir les bases et les principes de la recherche d'information, ses facteurs d'influences, ainsi que ses différents modèles ; le Système de Recherche d'information (SRI) où on analysera le fonctionnement d'un moteur de recherche, ses défis, ainsi que don évaluation ; le Traitement de Langage Naturel (TLN) où on va analyser les méthodes de traitement de langage naturel et ses utilités vis-a-vis de la recherche d'information ; et enfin le Modèle Vectoriel (Vector Space Model) qui est la méthode (modèle) choisi dans le cadre de ce mémoire où on va analyser ses principes, ses avantages ainsi que ses limites. La deuxième partie on va détailler la simulation et la réalisation du SRI proposé ci-dessus, ainsi que la liste de ces fonctionnalités et les outils de développement utilisé, et on analysera ses performances et ces limites. La dernière partie qui est la conclusion générale ou on va conclure notre travail de recherche ainsi de donner des perspectives.

Chapitre 1

Recherche d'information

1.1 Introduction

La recherche d'information est un domaine étendu et ancien qui trouve ses racines au début des années 1950, peu après l'invention de l'ordinateur (ZIANI, 2022). Plus précisément, elle date des années 1940 dès la naissance de l'ordinateur. Dans les années 1960 et 1970, la *RI* commence les expérimentations plus larges alors qu'au départ elle s'est concentré sur des applications des bibliothèques. Et puis l'intégration de l'IA dans la *RI* a partir des années 1980 ainsi que la naissance d'internet en 1990 propulse la *RI* et met en scène beaucoup d'applications. (Salton, 1989)

La *RI* suscite donc un grand intérêt parmi les chercheurs, devenant de plus en plus attrayante avec l'avènement de machines puissantes et performantes en matière de stockage, faciles à utiliser et fiables (Blair et Maron, 1985). La recherche d'information est désormais une tâche récurrente pour les utilisateurs, que ce soit sur smartphones, ordinateurs, etc., car elle répond à leurs besoins quotidiens (Bellaouar et al., 2009). Ce domaine traite des données non structurées, principalement des données textuelles comme le langage naturel (Baeza-Yates et Ribeiro-Neto, 1999). Citons quelques définitions de la Recherche d'Information :

Définition 1.1.1. Selon Gérard Salton, la Recherche d'Information (*RI*) consiste a structurer, analyser, organiser, stocker, et rechercher de l'information. Et aussi un activité de délivrer un ensemble des documents a l'utilisateur en fonction de son besoin d'information Bellaouar et al., 2009.

Définition 1.1.2. C'est la stockage, gestion, traitement, récupération des informations tel que des documents ou site web pour satisfaire un besoin informationnel de l'utilisateur (Abu-Salih, 2018).

Définition 1.1.3. La RI est un ensemble d'une base de documents et une phase de processus de recherche avec des méthodes, techniques permettant a l'utilisateur de récupérer un ou plusieurs

documents (Kien QUACHTAT, 2012).

Définition 1.1.4 (Système de Recherche d'Information)**.** On appelle SRI ou moteur de recherche, un ensemble de programmes informatiques, qui met en œuvre des techniques et moyens pour trouver les documents pertinents afin de satisfaire le besoin d'information de l'utilisateur (ZIANI, 2022).

Une courte histoire de la recherche d'information jusqu'a l'arrivé de Google en 1998, est illustré dans la Figure 1.1 et dans la Figure 1.2 le principe de base de la recherche d'information.

1.2 Information

1.2.1 C'est quoi une information ?

Une information est issu lorsqu'on donne un sens a un données. Un donnée est donc un unité élémentaire d'information, un octets constitué de bits c'est à dire des 0 et des 1. Un donnée est une représentation a laquelle une signification peur être rattaché, peut être quantitative ou qualitative et n'a pas de sens elle-même.

Une information est donc une collection des données pour donner une forme a une message que ce soit imagée, écrite ou orale pour réduire l'incertitude et transmettre quelque chose qui déclenche une action (Bellaouar, Djoudi et Zidat, 2009 ; Benayache, 2005).

Jérôme VELO (Jérôme VELO, 2009) définit une information comme un fait, une réalité, qui augmente la connaissance des individus. Elle est constitué de *données* et de *sens* qui est attribué par un individu. Le sens dépend de l'individu qui reçoit la données.

1.2.2 Pourquoi rechercher de l'information ?

Lorsqu'un utilisateur a un besoin informationnel (Besoin d'information) la RI est nécessaire pour satisfaire ce besoin. D'où l'initiation de sa Recherche d'information. C'est le besoin d'information de l'utilisateur qui déclenche alors le processus de RI qu'on va voir dans la Section 1.4.

Le défi de la recherche d'informations est de satisfaire le mieux la demande de l'utilisateur en retournant un ensemble des documents correspondant a ses demandes parmi un grand volume de documents (Mazari, 2022).

1.3 Document

1.3.1 C'est quoi un document ?

Bellaouar et al. propose quelques définitions d'un document, on va citer deux parmi ceux qui sont cité (Bellaouar et al., 2009).

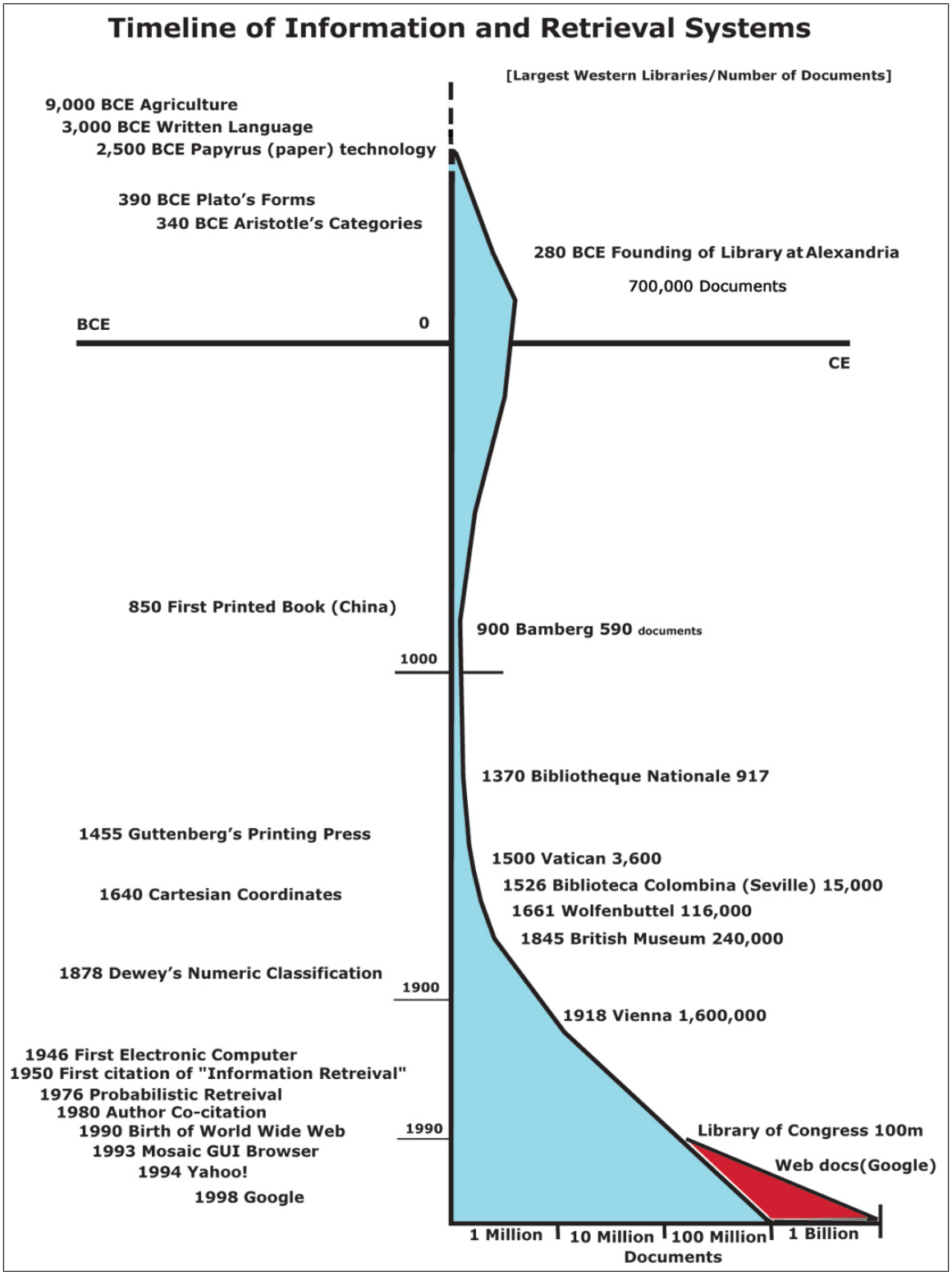

FIGURE 1.1 – Historique de la recherche d'information (DR. LOUNNAS BILAL, 2023)

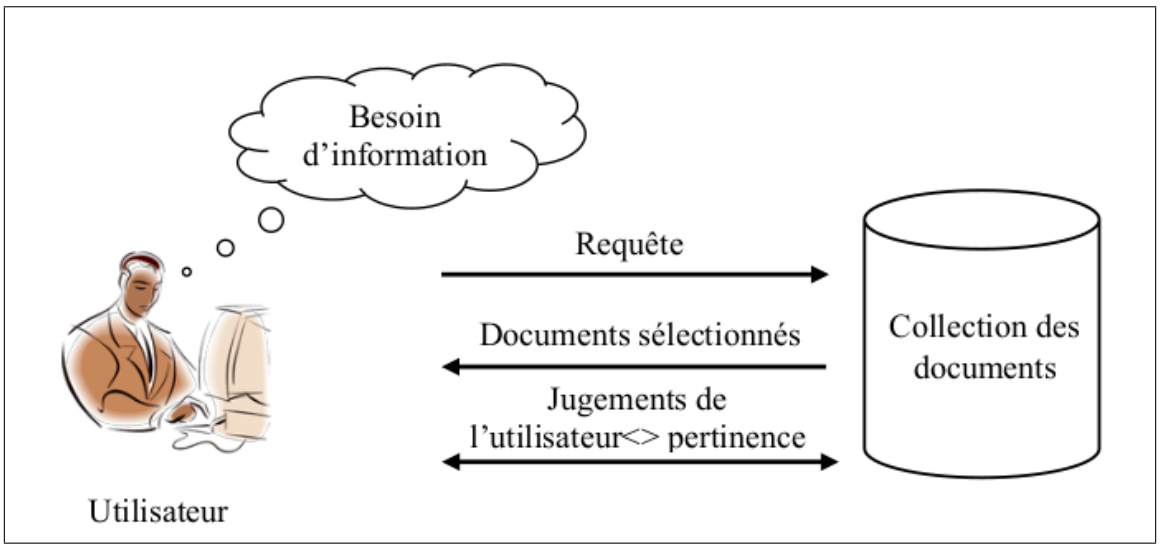

FIGURE 1.2 – Processus de base de la RI (MAZARI, 2022)

Définition 1.3.1. Un document est généralement l'expression d'une pensée humaine.

Définition 1.3.2. Un document représente toute base de connaissance fixé matériellement, susceptible d'être utilisé pour la consultation, l'étude ou la preuve (imprimé, manuscrit, représentation graphique, ...).

Définition 1.3.3. On appelle document toute unité qui peut constituer une réponse à une requête d'utilisateur. Un document peut être un texte, un morceau de texte, une page Web (HTML), une image, une bande vidéo, sons, etc (SALTON, 1989).

Un document composé de cinq éléments (BELLAOUAR et al., 2009) :

— **Support d'enregistrement** : papier, disque magnétique, CD/DVD-ROM, etc. C'est un support où le document est stocké.
— **Forme d'enregistrement** : papier, codage ASCII, codage vidéos, etc. C'est la forme de document sur le support.
— **Support de restitution** : papier, écran d'ordinateur, haut-parleurs, etc.
— **Forme physique de restitution** : encre sur papier, signal audio ou vidéos, etc.
— **Forme sémantique de restitution** : représentation qui respecte une certaine structure ou forme selon laquelle elle soit intelligible par le l'utilisateur. Écriture, sons, images, animés, etc.

Par la suite de ce document, nous traitons principalement des documents textuelles numérique, non structurés.

1.3.2 Classification des documents

1.3.2.1 Type de classification des documents

Pour pouvoir récupérer les documents qui sont similaires, ainsi que pour faciliter le filtre des résultats par l'utilisateur, il est important de classifier les documents.

La classification des documents permet donc de grouper ensemble les documents similaire dans un classe. Cette opération se fait sur la collection des documents ou corpus. Il y a deux types de classification (Baeza-Yates et Ribeiro-Neto, 1999) tel que la classification *local* et la classification *global* :

— **Classification global** : les documents sont groupés selon leur occurrences dans la collection.
— **Classification local** : les groupement des documents est affecté par le contexte défini par la requête et les documents local sélectionné. La classification varie selon la requête de l'utilisateur.

1.3.2.2 Algorithmes de classification global

Diverses algorithmes permettent de classifier les documents textuels, dont certains sont basé sur l'algorithme du *Machine Learning (ML)* et de l'*Intelligence Artificiel (IA)*. Citons quelques algorithmes présenté dans des revue scientifique des classifications des documents textuels (B S Harish, 2010 ; R Manikandan, 2018).

Pour l'approche du machine learning, il y a des algorithmes qui se base sur l'*apprentissage supervisé*, l'*apprentissage non supervisé* et l'*apprentissage par renforcement.*

— **K-Plus Proche Voisin ou K-Nearest Neighboor (KPP ou KNN)** : Est un approche non paramétrique, utilisé pour classifier des textes. L'approche définit K classe pour classifier. Pour décider si d_i appartient a la classe C_k, la similarité ou la dissimilarité pour tous les documents d_j dans le donnés d'entraînement est déterminé.
— **Arbre de décision** : La décision est basé sur certains conditions, et utilise un arbre. Les règles de classification sont représentés a travers la chemin du racine au feuille. L'algorithme d'arbre de décision le plus connu est *ID3* et ses successeurs tel que *C4.5* et *C5.* Cette approche a des avantages comme, simple a expliquer et a comprendre par des personnes non connaisseur du domaine, simple a mettre en place ainsi utile dans les analyses prédictives.
— **Naive Bayes Classifier** : Est un algorithme de machine learning, et généralement utilisé pour classifier les documents WEB, qui est un approche probabiliste. Permet de catégoriser des documents, des news, etc. Cette algorithme est efficace, et qui nécessite moins de données d'entraînement, ainsi le résultat est efficace.
— **Support Vector Machine** : Est une approche supervisé du Machine Learning. Le SVM Classifie les données dans différentes classe par la recherche du ligne qui sépare les données d'entrainement en classes, on l'appelle ligne d'hyperplan. Il y a deux catégories tel que le *SVM* linéaire et le *SVM* non linéaire. Cette approche est performant pour classifier les données d'entraînement ainsi corrige bien la classification pour les futures données.
— **D'autres algorithmes** : Il y a d'autres algorithmes pour classifier les documents, mais qui seront pas couvert dans ce travail.

1.4 Besoin d'information

Définition 1.4.1. C'est la besoin d'information qui déclenche la recherche d'information de l'utilisateur. C'est une sensation qui porterait un individu a s'engager dans une activité de recherche d'information (Bellaouar et al., 2009).

Définition 1.4.2. C'est une abstraction mentale dont l'utilisateur a besoin pour répondre a une question ou demande particulière et qui est exprimé en langage naturel (Mazari, 2022) (Reformulation nécessaire).

C'est cette besoin d'information que l'utilisateur doit traduire pour obtenir une requête afin qu'un SRI puisse satisfaire ce besoin, on dit souvent des mots clés de recherche (Baeza-Yates et Ribeiro-Neto, 1999).

La besoin d'information se catégorisent en trois types (Paradis, 1996) tel que :

- **Besoin vérificatif** : l'utilisateur recherche une donnée particulière, et sait comment y accéder pour vérifier le texte avec des données qu'il possède déjà. En d'autre terme, l'utilisateur possède déjà les données ou la partie mais il a besoin de vérification. Le besoin ne change pas au cours de sa recherche, on dit que c'est *stable*.
- **Besoin thématique connu** : l'utilisateur cherche a éclaircir, revoir ou trouver des nouvelles informations concernant un sujet ou domaine connu. La besoin peut changer (*stable*) ou non (*variable*) au cours de sa recherche ainsi qu'il peut se raffiner. Le besoin peut s'exprimer de façon incomplète.
- **Besoin thématique inconnu** : l'utilisateur cherche des nouveaux concepts ou relations hors des domaines ou sujet connus. La besoin est variable et toujours exprimé de façon incomplète.

Pour mieux élaborer ce processus, on va analyser dans la Section 1.8 la processus général de la recherche d'information, la processus en U.

1.5 Tâche de recherche de l'utilisateur

Une tâche de recherche c'est qu'un utilisateur doit faire pour satisfaire un besoin d'information (Boubée et Tricot, 2010 ; Kien QUACHTAT, 2012). Les tâches se catégorisent suivant deux types, une tâche *fermé* et une tâche *ouverte*. Une tâche fermé est une tâche dont l'utilisateur cherche une réponse exacte sur une requête donnée, tandis qu'une tâche ouverte est une tâche dont l'utilisateur cherche une réponse acceptable. Après certains études cités dans (Kien QUACHTAT, 2012) et d'autres chercheurs, que la tâche ouverte utilise d'avantage la navigation, tandis que la tâche fermé privilégie les moteurs de recherche.

Yang (Yang, 2000) cite deux stratégies de recherche en utilisant des requêtes :

- **Bottom-up ou mixtes :** la recherche commence par une sphère étroite (précise) puis l'élargir de plus en plus. L'utilisateur fait une recherche de plus précise a une recherche

plus générale.

- **Top-down :** la recherche commence par une sphère large (générale) puis le rétrécir de plus en plus. L'utilisateur fait une recherche de plus générale a une recherche de plus en plus précise.

Selon KIEN QUACHTAT (KIEN QUACHTAT, 2012), il y a trois type de tâches tel qu'une *tâche factuelle* qui est une tâche généralement fermé, une *tâche interprétative* qui est une tâche généralement ouverte et une *tâche d'exploitation* qui est une tâche complètement ouverte.

1.5.1 Tâche factuelle

Une tâche factuelle est catégorisé dans la tâche fermé, on peut le considérer comme une tâche fermé. Dans une tâche factuelle, les utilisateurs formulent de requêtes dans un SRI ou moteur de recherche, et lisent les résumés de chaque document dans la page de résultat. Ils veulent trouver une réponse sur la page de résultat et privilégie le résumé qui contient les mots clés, et en cliquant juste pour confirmation. Dans cette tâche les experts utilise plutôt la stratégie *bottom-up* tandis que les novices utilisent la stratégie *top-down*.

1.5.2 Tâche interprétative

Une tâche factuelle est catégorisé dans la tâche ouverte, on peut le considérer comme une tâche ouverte. Les utilisateurs commencent par le parcours des informations générales, puis affiner leurs recherche pour leurs besoins spécifiques. Ils privilégié la lecture de contenu des pages et la navigation, souvent une navigation interne a des pages web (via des liens internes) ou externes.

1.5.3 Tâche d'exploitation

Une tâche d'exploitation est une tâche complètement ouverte. Les utilisateurs se concentrent plus sur la page de résultat pour trouver une résultat qui semble correspondre au besoin, puis cliquent et naviguent dans la page pour explorer. Ils veulent trouver de l'information basé sur les contenues de la page. Dans cette tâche, les experts utilisent la stratégie *top-down* tandis que les novices utilisent la stratégie *bottom-up*.

1.6 Modèle de recherche de l'utilisateur

En général il y a deux approche de recherche d'information, dont l'approche classique qui est le *Ad-hoc* et et l'approche moderne qui est le *Filtering* (Filtre en français) (BAEZA-YATES et RIBEIRO-NETO, 1999).

1.6.1 Ad-hoc : approche classique

L'approche classique est l'approche la plus utilisée, d'où c'est une approche conventionnel de la Recherche de l'Information. Dans une approche classique, les documents sont resté relativement statique (les documents sont préparés a l'avance). Puis arrivé ensuite la requête des utilisateurs pour trouver une réponses dans ces documents. En d'autre terme, les documents restent statique jusqu'à ce que la requête de l'utilisateur arrive.

1.6.2 Filtering : approche moderne

L'approche moderne consiste a mettre en place un profile complexe de l'utilisateur pour savoir quel documents ou informations pourra lui interrelié. Dans cette approche, les requêtes (préférences, type d'informations qui lui intéressent, ...) de l'utilisateur restent statique jusqu'à ce qu'un nouveau document arrive dans le système, et que ce système décide si c'est pertinent par rapport au profil de l'utilisateur. Cette approche est souvent utilisé pour les news, marketing, ...

1.7 Facteur d'influence de la recherche d'information

La recherche d'information de l'utilisateur est influencé par plusieurs facteurs, tel que l'expérience, facteur socio-culturel, et d'autres facteurs. On peut citer d'après l'analyse de ces facteurs fait par Kien QUACHTAT (Kien QUACHTAT, 2012) les facteurs ci-dessous :

- **Expérience de l'utilisateur** : l'expertise du domaine implique la qualité de choix des mots clés a utiliser, la stratégie de recherche ; l'expertise du système implique la connaissance de fonctionnement d'un SRI et la stratégie de la RI ;l'expertise de la RI implique la capacité d'utiliser un SRI spécifique.
- **Tâche de recherche** : influencé par le type de tâche de l'utilisateur, qu'il recherche de réponse exacte (*tâche fermé*) ou recherche de réponse acceptable (*tâche ouverte*).
- **Facteur socio-culturel** : la contexte sociale et la culture est indissociable de la recherche d'information ; le genre influence aussi la recherche, la stratégie de recherche utilisé par les femmes est différente de celle de l'homme, et la chance de réussite est différente ; l'âge aussi a son influence, les jeunes et les personnes âgés ont différente façon de faire une recherche. Par exemple, les jeunes on tendance a consulter moins de pages et formuler des requête plus souvent que les âgées.
- **Connaissance de fonctionnement de SRI** : parfois les connaissances sont fausse par rapport aux capacités des moteurs de recherche. L'utilisateur estime qu'il sait utiliser un SRI alors qu'il n'exploite véritablement les capacités du SRI.
- **D'autres facteurs** : d'autres facteurs sont aussi détaillés dans la section 2.2.5, comme la limite de temps, caractéristiques individuelles. Un tableau de synthèse de ces facteurs est présenté dans la section 2.3.

Pour plus de détails sur ces facteurs d'influences, voir « Recherche d'information sur le WEB et moteurs de recherche: le cas des lycéens », section 2.2

1.8 Processus de recherche d'information

Le modèle de processus le plus utilisé est le modèle en U. C'est un modèle qui traite les documents et la requête séparément et faire une appariement après.

1.8.1 Processus en U

La processus en U se déroule en général quatre étapes (Baeza-Yates et Ribeiro-Neto, 1999) avec deux type de traitement, d'un coté la traitement des documents et de l'autre coté la traitement de la requête. L'*indexation des documents* : créer la base documentaires dans la base de données ; *traitement et opérations sur la requête* : traiter la requête de l'utilisateur, retirer les mots vides ; l'*appariement de la requête et les documents* : pour savoir quelles documents répond mieux a la requête de l'utilisateur ; Et enfin la *reformulation de la requête de l'utilisateur* : pour améliorer les résultats obtenues. La Figure 1.3 illustre le processus de la recherche d'information en U.

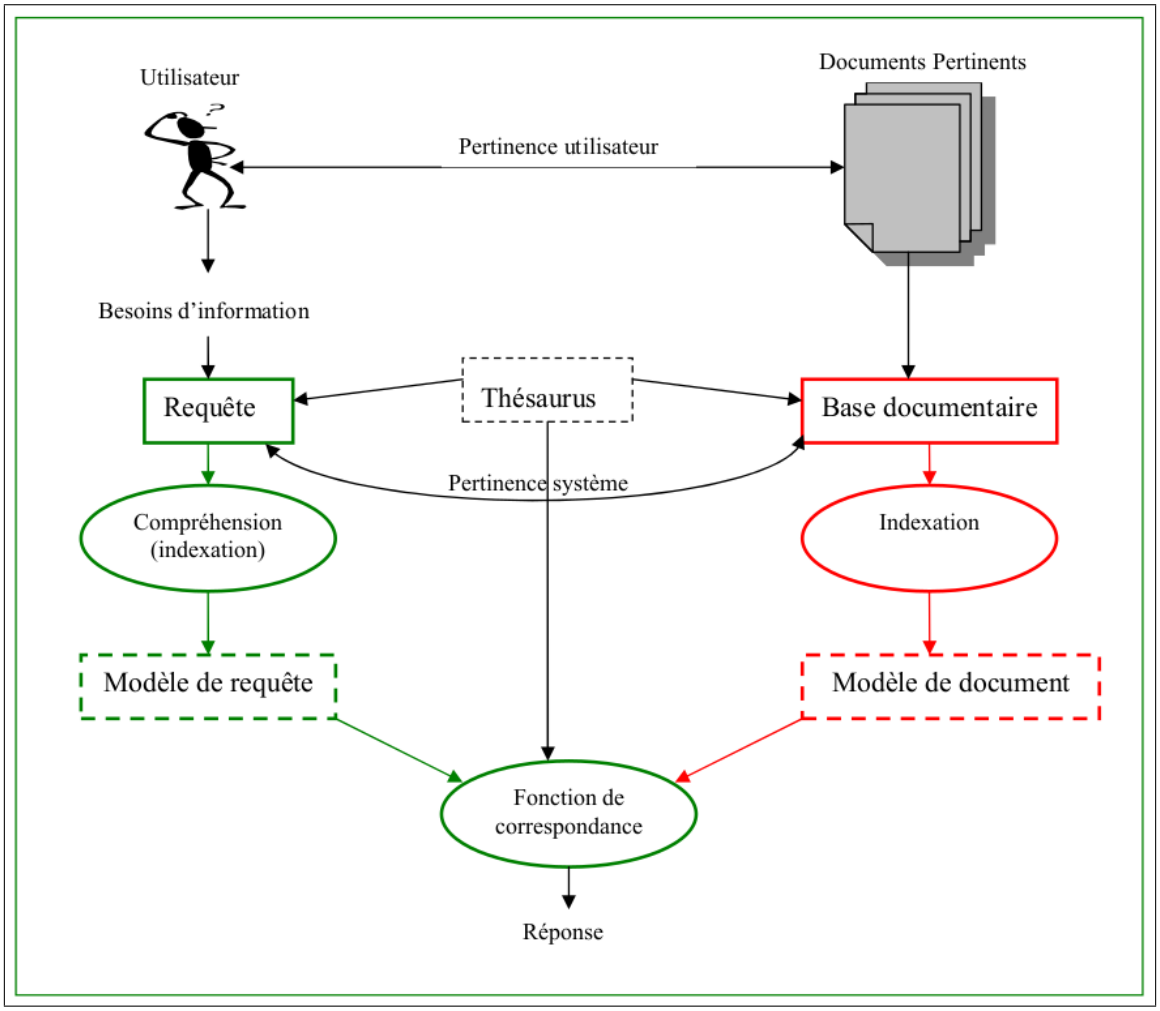

Figure 1.3 – Processus en U de la RI (Bellaouar et al., 2009)

Pour la suite, on va expliquer en détails chaque partie de ce processus pour mieux élaborer l'étape de la recherche d'information.

1.8.2 Première étape : Besoin d'information

Comme définit dans la Section 1.4, pour qu'une recherche d'information eu lieu, il faudra une besoin d'information. A ce stade la base documentaire est déjà mis en place et est déjà prête pour l'utilisation. L'utilisateur doit ensuite transformer son besoin en requête et obtenir des mots clés. Le système applique un traitement de texte (Traitement de Langage Naturel) sur la requête pour obtenir une requête finale qui sera utilisé pour l'appariement. La requête pourra contenir un ou plusieurs mots clés (Baeza-Yates et Ribeiro-Neto, 1999).

Définition 1.8.1 (Requête ou Query)**.** La requête est une interface entre l'utilisateur et le programme, formulé sous-forme des mots-clés par l'utilisateur, décrits en langage naturel (Mazari, 2022).

Définition 1.8.2 (Requête ou Query)**.** On appelle requête, l'expression du besoin d'information d'un utilisateur. Elle est en général exprimé sous fore des mots clés (Salton, 1989).

1.8.3 Indexation des documents

Pour travailler efficacement avec les documents et pour pouvoir rechercher des informations dedans, il faut représenter les documents sous formes des mots clés que contient le document, sous l'approche d'*indexation* (Yang, 2000).

Définition 1.8.3. L'indexation est l'action de représenter les contenu des documents sous forme d'index.

Définition 1.8.4. On appelle indexation ou processus d'indexation, le processus qui permet de construire les index a partir de l'analyse du documents tel que la vérification qu'un mot n'est pas dans un anti-dictionnaire (mots vides ou stop words), et la lemmatisation ou racinisation pour trouver la racine des mots. Cette approche est formalisé en utilisant un langage d'indexation. (Paradis, 1996).

1.8.3.1 C'est quoi un index ?

Un index est un mot ou groupe des mots sélectionné soit manuellement par un expert (personne expert du domaine en question), soit automatiquement par un ordinateur (programme informatique) pour représenter un document. Les termes d'indexation sont tous les index qui représentent un document. Un index est en général un nom ou groupe des noms.

Selon Salton, qui traite généralement les index comme représentant du document, Un index est une représentation synthétique de l'information relative a un document, qui met en évidence sa sémantique en vue d'une requête. Selon d'autres auteurs, se sont des mots clés (Paradis,

1996).

1.8.3.2 Approche d'indexation

Comme on a évoqué dans la Section 1.8.3.1 et cité dans (Yang, 2000) ainsi que dans (Mazari, 2022), il y a en général quatre approche d'indexation tel que l'*indexation manuel*, l'*indexation intelligent*, l'*indexation automatique*, et l'*indexation basé sur les métadonnées*. Il y a une approche *semi-automatique* qui est l'hybridation de l'indexation manuelle et l'indexation automatique, cette approche consiste a créer les index automatiquement puis ces index sont validés par un expert (Paradis, 1996).

- **Indexation manuelle (Human Indexing) :** c'est un expert humain qui fait l'indexation. Cette approche a une avantage d'être plus précise en sélection des termes d'indexation. Tandis qu'avec la croissance de volume d'informations, cette approche est de plus en plus obsolète, ainsi il se peut qu'il y a une incohérence entre les experts en sélectionnant les termes d'indexation. De plus cette approche est coûteux (Blair et Maron, 1985).
- **Indexation automatique (Automatic Indexing) :** c'est un programme informatique qui fait l'indexation. Cette approche a une avantage d'être moins coûteux que l'indexation manuel, et résout l'incohérence entre les termes d'indexation sélectionné. Mais peut être moins efficace en sélectionnant les index, les index peuvent être moins précise. Cette approche utilise généralement la Traitement de Langage Naturel (NLP) et de la Statistique (segmentation, suppression des mots vides, etc.).
- **Indexation intelligent (Intelligent Indexing / Agent-Based Indexing) :** utilisation des robots (Spiders / Crawlers) pour l'indexation du *WEB*. Un *robot* ou *spiders* est un programme informatique qui parcours les liens sur le web et index les documents trouvées. Cette approche a trois problèmes en général (Robot Standard Exclusion) tel que, les robots peuvent causer des *surcharge au niveau des serveurs*, ils sont *invasives*, et la *mise a jour fréquent des sites web*.
- **Indexation basé sur les métadonnées :** utilisation et interprétation des métadonnées, souvent utilisé dans la recherche d'information sur le web (HTML).

Dans le cadre de ce mémoire, on focalisera sur l'indexation automatique des documents textuelle par un programme informatique.

1.8.4 Analyse lexical ou Segmentation

L'analyse lexical des textes permet de faire une analyse de la lexique. Cette analyse peut varier suivant la langue utilisée. On analyse généralement les ponctuations, les espaces entre les mots et la casse des lettres. On considère que la séparation des mots est le caractère espace. L'analyse lexical est alors l'action de convertir les textes du document en groupe des mots (stream of words) qui peuvent être sélectionné comme termes d'indexation. Les nombres sont souvent une mauvaise index sauf s'il est mixé avec un mot. Les expressions régulières sont

souvent utiles pour extraire des nombres comme le numéro de carte de crédit, ... (Baeza-Yates et Ribeiro-Neto, 1999 ; ZIANI, 2022).

Ponctuation

Les ponctuations sont normalement supprimés dans le cas d'analyse lexicales. Sauf dans le cas pour distinguer 'x.id' et 'xid.', la ponctuation ne doit pas être supprimé. (Baeza-Yates et Ribeiro-Neto, 1999)

Casse des lettres

Il faut convertir tous les mots en majuscules ou en minuscules. Mais en particulier, les langages de commande (ligne de commande) de Linux, car l'utilisateur ne veut pas convertir la casse des lettres (Baeza-Yates et Ribeiro-Neto, 1999). Ça peut perdre aussi l'aspect sémantique après la conversation.

1.8.5 Élimination des mots vides (Stop words)

On appelle un mot vide, un mot qui est plus souvent utilisé dans un document ou dans les termes de recherche d'un utilisateur comme les prépositions, les articles, les conjonctions. Certains verbes, adverbes, et adjectifs pouvant être traité comme mots vides. Éliminer ces mots est important car un mot qui apparaît 80% dans le corpus n'est pas utile pour la recherche d'information. C'est l'action de supprimer les mots insignifiants (pronoms personnels, préposition, ...) (Baeza-Yates et Ribeiro-Neto, 1999 ; SAADOUNE, 2018 ; ZIANI, 2022).

Voici une liste non exhaustive des mots vides en français selon (Ranks NL, 2023) : le, de, du, mais, donc, car, ceci, cela, qui. A noter que pour un SRI Full-text qui prend tous les mots comme des termes d'indexations, il n'y a pas de suppression des mots vides (Baeza-Yates et Ribeiro-Neto, 1999).

1.8.6 Lemmatisation et Stemming

La lemmatisation ou racinisation de mot est une partie de le traitement de langage naturel (TLN) qui extrait la racine d'un mot pour regrouper les variants (biologiste, biologique → biologie ; suis, est → être). Cette approche réduit la taille des termes d'indexation (SAADOUNE, 2018 ; ZIANI, 2022).

Le Stemming regroupe aussi les variants d'un mot pour obtenir ce qu'on appelle un stem. Un stem est un portion d'un mot restant après la suppression de ses affixes (suffixes, préfixes), la suppression de pluriel, forme gérondif. Cette approche réduit aussi la taille des termes d'indexation. Le stemming ne permet pas de satisfaire la racinisation d'un mot (peut mal déterminer la racine), que certaine moteurs de recherche n'adopte pas le stemming (Baeza-Yates et Ribeiro-Neto, 1999 ; Vaibhav Singh et Vinay Singh, 2022).

Le stemming possède plusieurs techniques, mais on va citer les techniques principaux comme :

- **Table lookup** : regarder le stem dans un table, mais qui est moins pratique en terme de stockage et accessibilité.
- **Porter algorithme** : est une technique simple, intuitive, peut être implémenté efficacement. Cette approche utilise la suppression des affixes.
- **Successeur** : est un approche plus complexe que la suppression des affixes.Se base utilisation des connaissances linguistiques structurés pour identifier les morphes.
- **N-Grams** : est une approche orienté plus vers la classification que stemming. Cette approche se base sur l'identification des diagrammes et trigrammes.

D'autres méthodes sont disponibles actuellement comme celle qui se base sur les expressions régulières, ...

1.8.7 Normalisation

La normalisation permet de classer les termes comme équivalent. Par exemple, le mot U.S.A doit être traité de la même façon que USA.Cette approche de normalisation peut amener a des problèmes (SAADOUNE, 2018).

1.8.8 Sélection et pondération des termes

Comme on dit précédemment, pas tous les mots dans le documents sont significatifs. Sauf pour un moteur de recherche *full-text* qui index tous les termes. Et aussi pas tous les mots on une même degré ou poids d'importance dans un document. D'où la pondération des termes. Pour déterminer le poids d'un terme dans un collection des documents, on utilise en général deux approche : *Term Frequency (TF)* et *Inverse Document Frequency (IDF)*.

Le TF (Term Frequency) du terme t dans un document d est le nombre de fois où le terme t apparaît dans le document d. C'est la fréquence du ter t dans le document d. Plus ce fréquence est important, plus le terme est important pour représenter le document. Cette mesure sera normalisé afin d'éviter d'avoir un poids trop significatifs pour les documents avec beaucoup de contenu, on l'appelle *Normalized Term Frequency*.

L'IDF (Inverse Document Frequency) du terme t dans la collection des documents D est le nombre de document dans la collection contenant le terme t. C'est le nombre de document où le terme t apparaît dans la collection des documents D. Cette approche permet de déterminer l'unicité d'un terme au sein de la collection des documents. Un terme est important si moins de documents la contiennent ; un terme qui apparaît dans presque tous les documents n'est pas significatif, et peut être traité comme un mot vide.

La troisième approche qui est la combinaison des deux est la *TF-IDF*. La TF-IDF est la

multiplication des deux valeurs obtenues précédemment ($TF * IDF$) pour obtenir le poids final du terme. Le terme est important s'il apparaît dans peu de documents (IDF) mais qu'il apparaît plusieurs fois dans le document qui le contient (TF).

Une fois ces termes sélectionné avec les poids correspondant, on crée un index inversé ou fichier inversé pour stocker ces termes. Cet index inversé va faire correspondre le document identifié par un ID avec les termes qui le compose avec le poids correspondant (Abu-Salih, 2018 ; Baeza-Yates et Ribeiro-Neto, 1999 ; SAADOUNE, 2018 ; Vaibhav Singh et Vinay Singh, 2022). Tous ces notions va être détaillé un peu plus bas dans le document lorsqu'on abordera le modèle utilisé dans le cadre de ce mémoire.

1.8.9 Étape d'appariement

Pour pouvoir satisfaire le besoin de l'utilisateur, il faut apparier la requête de l'utilisateur et les documents (fichier inversé) en utilisant un algorithme d'appariement. Ces algorithmes permet de calculer la similarité entre un document et une requête et savoir quels sont les documents qui correspondent le mieux avec la requête avec un degré de similarité. Mathématiquement, c'est calculer la distance euclidien entre la requête et le document.

On calcule ces mesures pour déterminer la pertinence entre la requête de l'utilisateur et les documents dans le corpus, on l'appelle *pertinence système*. Il y a aussi le jugement qui provient de l'utilisateur en vue des résultats qu'il obtient par rapport a sa demande, on l'appelle *pertinence utilisateur* (Martinet, 2004).

La notion de pertinence est très complexe. En général un document est pertinent si l'utilisateur trouve les informations qui satisfait ses besoins dans le document. C'est sur ce notion de pertinence que le SRI doit juger la pertinence. (Salton, 1989)

Les documents trouvés sont retournés a l'utilisateur généralement suivant un degré de pertinence décroissante par rapport a la requête. Certains modèle ne possède pas ce classement des résultats comme le modèle Booléen qu'on va voir dans la section précédente, ainsi que décortiquer ces mesures de similarité.

1.8.10 Reformulation de la requête

La requête de l'utilisateur est parfois courte ou incomplète, alors il faut reformuler la requête par l'ajoute, modification ou suppression des mots clés dans la requête. Il est aussi possible de modifier le poids des termes dans la requête. Parmi les différentes approches proposé, Mazari (Mazari, 2022) note trois approches utilisé dans la reformulation de la requête tel que :

— **Thésaurus** : cette approche utilise une base de connaissance linguistiques pour améliorer la requête de l'utilisateur.
— **Basé sur les co-occurrences** : utilisent le calcul basé sur des co-occurrences des termes.

— **Relevance feed-back** : cette approche se base sur la reformulation de la requête par l'utilisateur par rapport aux pertinences des documents qu'ils reçoivent. En rajoutant, modifiant ou enlevant des termes de recherche ou mots clés.

1.9 Modèle de RI

1.9.1 Introduction

Il y a deux raisons pour avoir un modèle dans la recherche d'information. La première, un modèle guide la recherche et donne sens a la discussion académique. La seconde, un modèle est utilisé comme un plan (*blueprint*) pour implémenter un système de recherche d'information (Hiemstra, 1999).

Un modèle sert a décrire le processus computationnel du RI (comment les documents sont stockés, comment sont stocké les index, etc.), décrire le processus humain (besoin d'information, interaction), ainsi la définition implicite ou explicite de la pertinence (A. Rozenknop, s. d.).

Pour bien formaliser la recherche d'information, il est nécessaire de définir des modèles a utiliser. L'étape d'appariement dans la Section 1.8.9, nécessite un modèle.

1.9.2 Caractéristique formel (DQFR)

Définition 1.9.1. Un modèle de recherche d'information est un quadruplet *DQFR* (Baeza-Yates et Ribeiro-Neto, 1999 ; Vaibhav Singh et Vinay Singh, 2022) tel que :

— **D** : ensemble de représentation (vue logiques) des documents dans la collection.
— **Q** : ensemble de représentation (vue logiques) des besoins de l'utilisateur. On l'appelle requête ou query en anglais.
— **F** : framework pour modéliser les documents, la requête ainsi que leurs relations.
— **R** (q_i, d_j) : fonction de classement qui associe un nombre réel avec la requête $q_i \in Q$ et le document $d_j \in D$. Cette classement permet d'ordonner les résultats selon leur pertinence.

1.9.3 Taxonomie de modèle

Selon Baeza-Yates et Ribeiro-Neto il y a quatre familles principales des modèles de RI qui est illustré dans la Figure 1.4 :

— **Les modèles basé sur les textes du document** : les modèles de RI classique (*Théorie des ensembles, Algébrique, Probabiliste*), et les modèles basé sur le texte semi-structuré (*Texte semi-structuré*)
— **Les modèles basé sur le liens entre documents** : les modèles orienté web (*PageRank, Hubs et autorités*)

— **Les modèles basé sur les documents multimédia** : les modèles de recherche d'images, vidéos, musiques, etc.

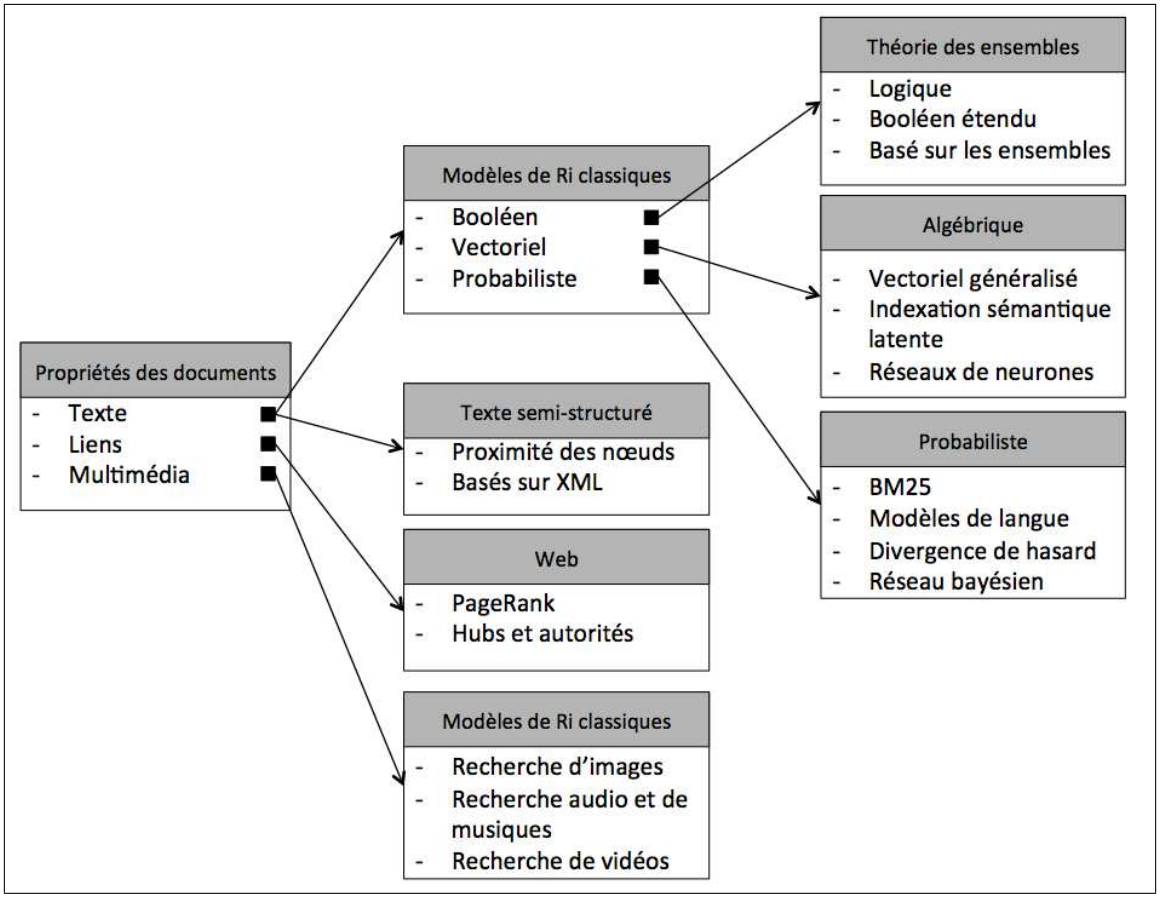

FIGURE 1.4 – Taxonomie de modèle de RI (SOULIER, 2014)

Dans le cadre de ce mémoire, on va analyser la famille des modèles classiques basé sur les textes du document.

1.9.4 Catégorie de modèle

Dans cette famille (BAEZA-YATES et RIBEIRO-NETO, 1999 ; SOULIER, 2014 ; ZIANI, 2022), il y a trois grandes catégories des modèles : le modèle *théorique ou ensembliste* (theoretic), le modèle *algébrique* (algrebraic) et le modèle *probabiliste* (probabilistic).

Dans le modèle ensembliste, on trouve principalement le modèle *Booléen*, le modèle *Booléen étendue*. Dans le modèle algébrique, on trouve principalement le modèle *vectoriel* (Vector Space Model), le modèle de *réseau de neurone* (Neural Network Models), et d'autres modèles. Dans le modèle probabiliste, on trouve principalement le modèle probabiliste (Probabilistic model), le *réseau d'inférence* (Inference Network) et d'autres modèles.

Ces modèles sont des modèles de recherche, mais il y a aussi des modèles pour la navigation (Browsing) (BAEZA-YATES et RIBEIRO-NETO, 1999).

1.9.5 Modèle ensembliste

1.9.5.1 Modèle Booléen (Boolean Model)

Un exemple de requête dans le modèle Booléen est illustré sur la Figure 1.5.

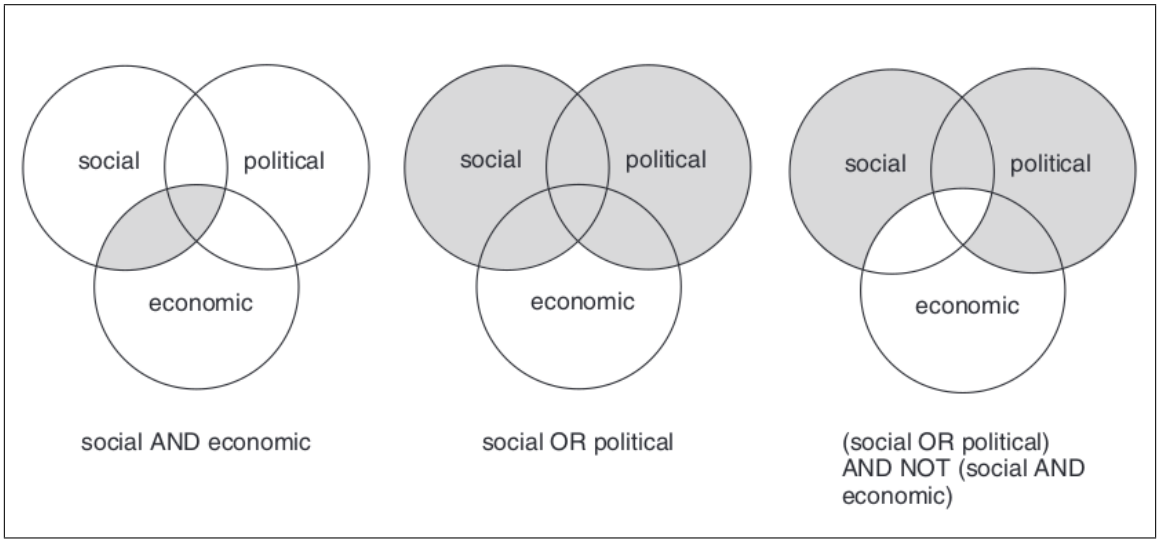

FIGURE 1.5 – Exemple de diagramme de requête (HIEMSTRA, 1999)

C'est un modèle de base, et le premier modèle utilisé dans la recherche d'information. Dans ce modèle, les documents et la requête sont représentés sous forme des mots clés. Ainsi la requête est exprimé en utilisant les opérateurs booléens *ET (AND)*, *OU (OR)* et *NON (NOT)*. La stratégie de recherche est basé sur la décision binaire. Tous les termes d'indexation ont la même poids et qui sont pondérés de façon binaire, soit il existe dans le document soit non. S'il existe dans le document, son poids est 1, sinon 0. La similarité entre q et d_j est égal a 1 si le document d_j valide l'expression booléen q, sinon 0. Un document est donc jugé pertinent par le système s'il valide l'expression booléen de la requête.

Ce modèle a des avantages d'être facile a mettre en place, simple (utilise l'algèbre de Boole) et exacte au niveau des résultats. Ce pour cette exactitude que ce modèle est aussi un modèle exacte (Exact Match Model).

Par contre, ce modèle a des lacunes, il est difficile de transformer le besoin d'information en expressions booléen surtout pour un *utilisateur lambda* (un utilisateur qui ne sait absolument rien de l'expression booléen) donc la requête a tendance un peu simplifié ou même incomplète ce qui affecte la qualité des résultats. Et puisque la décision est binaire, et que le poids soit 1 soit 0, il n'y a pas de classement possible pour les documents retournés. L'utilisateur est alors obligé de parcourir les résultats pour juger la pertinence. Puisque le modèle est exacte, le modèle ne permet pas de sélectionner les documents qui peuvent intéressé l'utilisateur et qui peuvent être pertinent par rapport a ses requête, et qu'il est impossible de récupérer une réponses partielles qui valide certains expression de la requête. (BAEZA-YATES et RIBEIRO-NETO, 1999 ; SOULIER, 2014).

Définition 1.9.2 (Modèle booléen)**.** On note $w_{ij} \in \{0, 1\}$ le poids du terme numéro i dans le document numéro j qui est binaire $\vec{q}_{dnf}$, forme normal de disjonction de la requête q. Considérons $\vec{q}_{cc}$ composant conjonctive de $\vec{q}_{dnf}$. La similarité du document d_j par rapport a la requête q est définie par :

$$Sim(d_j, q) = \begin{cases} 0 & Sinon \\ 1 & S'il\ \exists \vec{q}_{cc} \mid (\vec{q}_{cc} \in \vec{q}_{dnf}) \bigwedge (\forall k_i, g(\vec{d_j}) = g_i(\vec{q}_{cc})) \end{cases}$$

Si $Sim(d_j, q) = 1$, le document d_j est pertinent a la requête q ; dans le cas contraire, le document n'est pas pertinent par rapport a la requête.

1.9.5.2 Modèle Booléen Étendu (Extended Boolean Model)

En vue de ces lacunes, le modèle booléen est amélioré afin de résoudre certains problèmes avec le modèle Booléen. Premièrement, le modèle Booléen étendue a mis en place la possibilité de faire une similarité partiel et la pondération des termes. Il est étendu en ajoutant la fonctionnalité du modèle vectoriel qui est la combinaison de formulation de requête avec les caractéristiques du modèle vectoriel. Ce modèle est introduit par Salton, Gerard et Fox, Edward A et Wu, Harry (Salton, Gerard et Fox, Edward A et Wu, Harry, 1983) en 1983. Mais ce modèle n'a pas été largement utilisé (Baeza-Yates et Ribeiro-Neto, 1999).

1.9.5.3 D'autres modèles

Il y a aussi d'autres modèles dans la catégorie ensembliste, tel que le modèle de Fuzzy (Fuzzy Set Model, Region Models), ..., mais qui ne sont pas détaillé dans le cadre de ce mémoire. Pour plus de détails, voir chapitre 2 (Baeza-Yates et Ribeiro-Neto, 1999) et chapitre 1 (Hiemstra, 1999).

1.9.6 Modèle algébrique

1.9.6.1 Modèle Vectoriel (Vector Space Model)

Ce modèle a un alternative qui est le *Modèle Vectoriel Généralisé* par Salton, Gerard et Wong, Anita et Yang, Chung-Shu (Salton, Gerard et Wong, Anita et Yang, Chung-Shu, 1975), qui se base sur des vecteurs. Les documents et la requête sont représentés par un vecteur dans un *t-espace vectoriel* (représentation algébrique). Ce modèle accorde une appariement partielle, et les termes ne sont pas pondérés de manières binaires. Les pondérations des termes le plus souvent utilisé sont la *TF*, *IDF*, et la *TF-IDF* (SAADOUNE, 2018).

Un score de similarité est alors exprimé comme un mesure de proximité entre deux entités correspondant a l'angle qui sépare les deux vecteurs. Ces mesures sont présentés dans la Section 1.10.

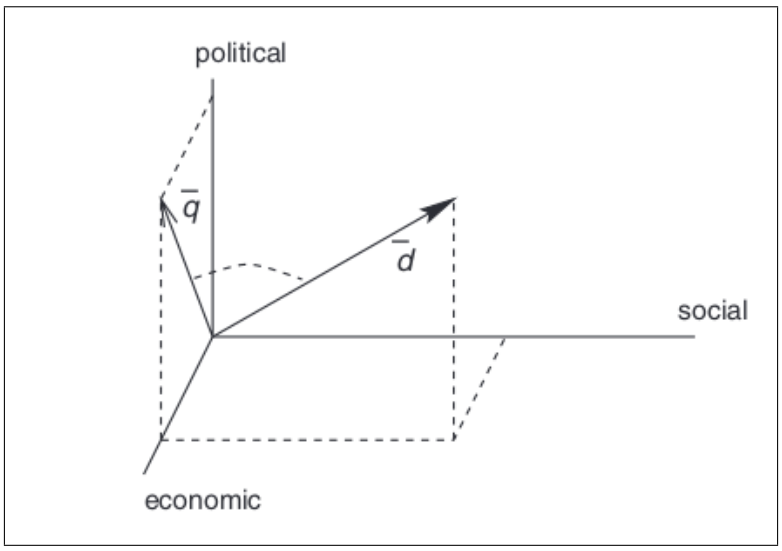

FIGURE 1.6 – Représentation du document et la requête (HIEMSTRA, 1999)

Le modèle vectoriel a certains avantages comme la facilité mise en œuvre, la possibilité d'avoir un appariement approximatif avec un degré de pertinence (l'utilisateur reçoit des documents qui pourrait lui intéressé), la possibilité d'organiser les résultats suivant leur pertinence (l'utilisateur passe moins de temps a filtrer les résultats puisqu'ils sont déjà ordonné), ainsi le pouvoir de définir une limité pour la mesure de similarité et de n'afficher que les documents qui sont en dessus de cette limite pour éliminer les résultats le moins pertinent. Ce modèle est populaire, et le plus utilisé par les moteurs de recherche actuel. (BAEZA-YATES et RIBEIRO-NETO, 1999 ; SOULIER, 2014 ; ZIANI, 2022)

Par contre, l'indépendance des termes d'indexation implique la perte de la notion de sémantique du document. Mais ce problème a été solutionné par la mise place de regroupement des termes qui ont la même sens, on l'appelle *N-grammes*. Ou bien une autre approche est d'utiliser le modèle d'*indexation sémantique* latente (Latent Semantic Index).

Définition 1.9.3 (Modèle Véctoriel)**.** On note w_{ij} le poids positif et non binaire, du terme i dans le document j qui est associé avec un pair (k_i, d_j) et w_{iq} le poids du terme i dans la requête q. La requête est définie par le vecteur : $\vec{q} = (w_{1q}, w_{2q}, \ldots, w_{tq})$ où t le nombre total des termes d'indexation dans le système. Un document d_j est présenté par le vecteur : $\vec{d_j} = (w_{1j}, w_{2j}, \ldots, w_{tj})$.

La calcul de similarité entre ces deux vecteurs se traduit par la formule :

$$Sim(\vec{d}, \vec{q}) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t}(w_{i,j})^2} \times \sqrt{\sum_{i=1}^{t}(w_{i,q})^2}}$$

Avec $|\vec{d_j}|$ et $|\vec{q}|$ norme du vecteur document et de la vecteur requête.

1.9.6.2 Autre modèle

Il y a d'autres modèles dans la catégorie algébrique tel que le modèle d'Indexation Sémantique Latente (Latent Semantic Indexing) et le modèle de réseau de neurones (Neural Network Model).

L'approche d'indexation sémantique latente est introduit en 1988, qui permet de garder la sémantique des documents pour avoir une relation entre les termes d'indexation. Cette approche se base sur l'appariement de contexte au lieu d'index, voir *Modern Information Retrieval*, Page 45 (Baeza-Yates et Ribeiro-Neto, 1999) pour les détails complet.

Le modèle neuronale se base sur des processus d'activation des neurones (spread activation process), cette approche est plus orienté vers l'intelligence artificielle (Baeza-Yates et Ribeiro-Neto, 1999).

1.9.7 Modèle probabiliste

Dans le modèle probabiliste, le framework pour modéliser les documents et la représentation de la requête se base sur la théorie des probabilités. Les détails seront présentés dans *Modern Information Retrieval*, Section 2 (Baeza-Yates et Ribeiro-Neto, 1999).

1.9.7.1 Modèle probabiliste

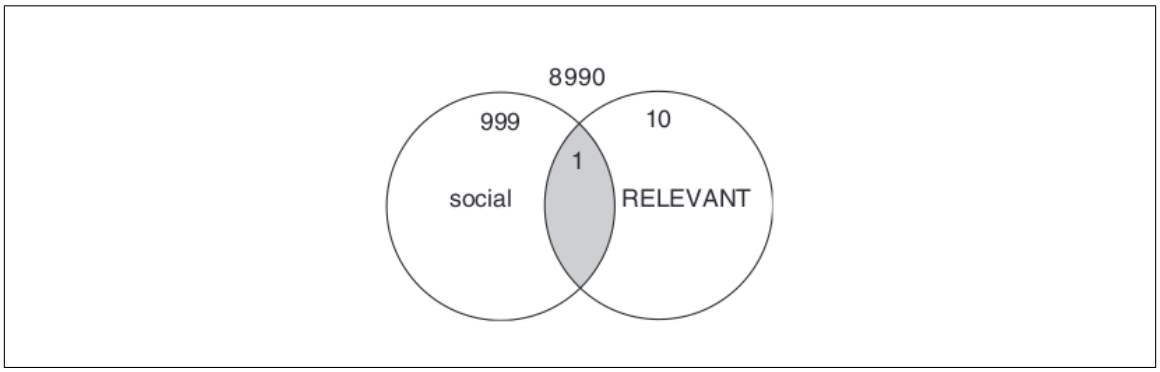

Figure 1.7 – Diagramme de collection pour le terme *social* (Hiemstra, 1999)

Le principe du modèle probabiliste est de donner la requête q de l'utilisateur et un document d_j dans la collection. Puis le modèle estime la probabilité que l'utilisateur trouve le document d_j pertinent. Le modèle assume que ce probabilité de pertinence dépend seulement de la requête et la représentation du document. La mesure de similarité utilise la *formule de Bayes* (Bayes's rule). Les détails sont présentés dans *Model de recherche d'information*, Section 2 (Hiemstra, 1999) et *Modern Information Retrieval*, Section 1.4 (Baeza-Yates et Ribeiro-Neto, 1999).

Définition 1.9.4 (Modèle probabiliste)**.** On note $w_{ij} \in \{0, 1\}$ le poids du terme i dans le document j, et $w_{iq} \in \{0, 1\}$ le poids du terme i dans la requête q.
R : Ensemble des documents connu ou initialement estimé d'être pertinent.
$\bar{R}$: Complément de R, l'ensemble de documents non pertinent.

$P(R/d_j)$: Probabilité de pertinence du document d_j pour la requête q.
$P(\bar{R}/d_j)$: Probabilité de non pertinence du document d_j pour la requête q.

$$Sim(d_j, q) = \frac{P(R/\vec{d_j})}{P(\bar{R}/\vec{d_j})} = \frac{P(\vec{d_j}/R) \times P(R)}{P(\vec{d_j}/\bar{R}) \times P(\bar{R})}$$

Où :
$P(d_j/R)$: Probabilité de sélection du document d_j au hasard provenant de l'ensemble R des documents pertinents.
$P(R)$: Probabilité qu'un document sélectionné au hasard dans la collection entière est pertinent.

Or, $P(R)$ et $P(\bar{R})$ est la même pour tous les documents, alors la formule de similarité devient :

$$Sim(d_j, q) \sim \frac{P(\vec{d_j}/R)}{P(\vec{d_j}/\bar{R})}$$

Avec l'application des poids de terme, la formule de similarité final est donc :

$$Sim(d_j, q) \sim \sum_{i=1}^{t} (w_{iq} \times w_{ij}) \times \left(\log \frac{P(k_i/R)}{1 - P(k_i/R)} + \log \frac{P(k_i/\bar{R})}{1 - P(k_i/\bar{R})} \right)$$

Tel qu'au départ :
$P(k_i/R)$: 0.5
$P(k_i/R) = \frac{n_i}{N}$ avec n_i le nombre de documents qui contient le terme k_i, et N le nombre total des documents.

Ce modèle a un avantage d'avoir le classement des résultats suivant leur probabilité d'être pertinent dans l'ordre décroisant. Par contre, il a quelques lacunes tel que le besoin de savoir initialement la séparation des documents en ensembles pertinent et non pertinent, le modèle ne prend pas en compte le fréquence de terme dans un document ce qui implique que tous les poids des termes sont binaire, ainsi les termes d'indexation sont indépendants (Baeza-Yates et Ribeiro-Neto, 1999).

1.9.7.2 Autres modèles

Il y a d'autres modèles dans ce catégorie, qui ne sont pas détaillé dans le cadre de ce devoir, comme le *Réseau d'Inférence* (Inference Network), *Réseau Bayésien* (Bayesian Network), le modèle de *Poisson* (The 2-Poisson model) (Hiemstra, 1999).

1.10 Mesure de similarité

Citons quelques méthode de similarité (Abu-Salih, 2018 ; SAADOUNE, 2018 ; Vaibhav Singh et Vinay Singh, 2022) :

- **Produit scalaire (Dot Product)** : c'est le produit scalaire entre deux vecteurs. La requête et les documents sont représenté par des vecteurs de même dimension.
- **Similarité cosinus (Cosine Similarity)** : c'est la mesure le plus populaire. Utilise le produit scalaire ainsi que l'angle formé entre les deux vecteurs (document et requête)
- **Coefficient de Jaccard (Jaccard Coefficient)** : mesure statistique de similarité entre des collections (sample set)
- **Coefficient de Dice (Dice Coefficient)** : même que celle de Jaccard, mais double le poids de vecteur

Puisque nous nous focaliserons sur le modèle vectoriel dans le cadre de ce mémoire, les quatre principaux mesures de similarité présenté dans la Section 1.8.9 sont les mesures utilisés dans le modèle vectoriel. Ces mesures sont utilisé pour déterminer la pertinence système c'est a dire la pertinence des documents par rapport a la requête de l'utilisateur. A bien noter que tous les poids des termes d'indexation sont normalisé avec TF-IDF avant la calcul de ces mesures. Ces mesures sont optionnelles, mais la mesure de similarité le plus populaire est celle de *cosinus* (Vaibhav Singh et Vinay Singh, 2022).

La pondération des termes se traduit alors comme suit :

$$d_k = q_k = tf(k,d) \cdot \log \frac{N}{df(k)}$$

Avec : Term Frequency (TF) :

$$TF(k,d) = \frac{Nombre\ d'occurences\ du\ terme\ k\ dans\ le\ document\ d}{Nombre\ total\ des\ termes\ dans\ le\ document\ d}$$

Inverse Document Frequency (IDF) :

$$IDF(t,D) = \log \frac{Nombre\ total\ des\ documents\ dans\ le\ corpus\ N}{Nombre\ de\ documents\ qui\ contient\ le\ terme\ t}$$

Term Frequency-Inverse Document Frequency (TF-IDF) :

$$TF - IDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

1.10.1 Produit scalaire

Le produit scalaire ou Dot product (Inner Product) en anglais est la mesure de base, qui est utilisé dans d'autres mesure comme celle de cosinus, Jaccard ainsi que Dice. Cette mesure calcul le produit scalaire entre le vecteur document et la vecteur requête. La formule pour calculer ce mesure est ci-dessous :

$$InnerProduct(\mathbf{A}, \mathbf{B}) = \mathbf{A} \cdot \mathbf{B}$$

1.10.2 Similarité cosinus

La mesure cosinus ou Cosine Similarity est la mesure la plus populaire. Cette mesure utilise le produit scalaire en introduisant l'angle formé par les deux vecteur (document et requête). La formule pour calculer cette mesure est ci-dessous :

$$Sim(\vec{d}, \vec{q}) = \frac{\vec{d} \cdot \vec{q}}{\|\vec{d}\| \cdot \|\vec{q}\|} = \frac{\sum_{k=1}^{n} d_k \cdot q_k}{\sqrt{\sum_{k=1}^{n} (d_k)^2} \cdot \sqrt{\sum_{k=1}^{n} (q_k)^2}}$$

1.10.3 Similarité de Jaccard

La mesure de Jaccard ou Jaccard's Similarity utilise aussi le produit scalaire, c'est une mesure basé sur la statistique. La formule pour calculer cette mesure est ci-dessous :

$$Jaccard\ Similarity(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

1.10.4 Similarité de Dice

La *mesure de Dice* ou *Dice's Similarity* est la mesure utilise le produit scalaire. La formule pour calculer cette mesure est ci-dessous :

$$Dice\ Similarity(A, B) = \frac{2|A \cap B|}{|A| + |B|}$$

1.11 Conclusion

La recherche d'information (RI) est une domaine vaste, intéressant mai aussi a des défis comme pour la partie indexation, etc. Elle est ancienne mais a connu beaucoup d'évolution, avec l'explosion de volume d'information disponible sur internet par le biais des réseau sociaux, des blogs, ainsi qu'avec l'arrivée de l'intelligence artificielle ainsi que del la Machine Learning (ML).

Elle propose différentes modèles mathématiques qui sont adaptés suivant les types de document (textuelle, image, etc.) pour formaliser la recherche, pour définir la mesure de pertinence entre le document et la requête, qui pourront être améliorer en utilisant les algorithmes de l'intelligence artificielle.

Dans la partie suivante, on va analyser comment ces modèles seront utilisé a travers un Système de Recherche d'information.

Chapitre 2

Système de recherche d'information

2.1 Introduction

Citons d'abord quelques définitions d'un SRI tiré de quelques travaux de recherches, ainsi on va analyser ses composants et puis on va analyser sa performance en utilisant des mesures de performance.

Définition 2.1.1. Un système de recherche d'information (SRI) est un système (ensemble des programmes informatiques) qui permet de retrouver les documents pertinents à une requête d'utilisateur en mettant et œuvre des moyens et des techniques, à partir d'une base de documents volumineuse. La requête de l'utilisateur est souvent composé des mots clés (Salton, 1989 ; ZIANI, 2022).

Définition 2.1.2. On appelle Système de Recherche d'Information (SRI) un ensemble des programmes, qui sert a interfacer avec l'utilisateur, de prendre les requêtes et de les interpréter afin de faire une recherche dans l'index, appliquer un modèle d'appariement et retourné les documents jugés pertinents a l'utilisateur (Mazari, 2022).

Définition 2.1.3. Un Système de Recherche d'Information (SRI) permet de récupérer la requête de l'utilisateur, de l'analyser, puis rechercher dans la base documentaire les documents qui correspondent (pertinent) a la requête et de retourner les résultats a l'utilisateur (Vaibhav Singh et Vinay Singh, 2022).

Définition 2.1.4. Un moteur de recherche est un outil (logiciel) qui permet de collecter des informations, rechercher des informations spécifique sur le Web ou sur un ordinateur personnel, à partir de mots clés et présente les résultats a l'utilisateur (Journal du Net, 2023 ; Mozilla Developer Network, 2023).

2.2 Objectif et efficacité

En général il y a deux catégorie de problème d'un SRI, tel que le problème *centré utilisateur* comme le problème lié a la compréhension et l'analyse de comportement de l'utilisateur ; et le problème *centré système* comme le problème lié a la création d'index efficace, traitements des requêtes avec plus de rapidité et efficacité ainsi que le développement d'un algorithme de classement pour améliorer les résultats (Baeza-Yates et Ribeiro-Neto, 1999).

Mais le problème central d'un SRI est *centré système* qui est de savoir comment satisfaire la besoin d'information de l'utilisateur le plus rapide possible et efficace. C'est un problème plus délicat a résoudre puisqu'il y en a plusieurs facteurs relié a ces performances. De l'autre coté il y a des recherches qui font des analyses de comportement des utilisateurs, ses impactes sur la recherche d'information ainsi que l'utilisation d'un SRI. Parmi ces différentes études, il y a celle qui est traité par Kien QUACHTAT (Kien QUACHTAT, 2012) dans le cadre des lycéens.

L'objectif d'un SRI est donc de localiser et retourner les documents pertinents par rapport a une requête pour répondre et satisfaire le besoin d'information de l'utilisateur. Le SRI relève donc le défi de rechercher des documents pertinents par rapport au requête de l'utilisateur parmi un très grand volume d'informations (Mazari, 2022).

L'efficacité d'un SRI est jugé par la capacité de comprendre ce que l'utilisateur veut, ce que l'utilisateur demande a partir de la requête. Ainsi sur sa capacité de fournir les documents pertinents et les classer suivant leurs pertinences par rapport a la requête de l'utilisateur (Vaibhav Singh et Vinay Singh, 2022). Et aussi la rapidité de récupération des informations et l'accessibilité au niveau de l'interface utilisateur (Yang, 2000).

Dans le cadre de ce devoir, on se focalisera plus sur la performance relié au problème centré système que ceux qui est centré utilisateur.

2.3 Élément principaux

Un Système de Recherche d'Information a deux éléments principaux, tel que la *base de donnée* et l'*algorithme de classement* (ZIANI, 2022). Ces deux éléments sont indispensables pour un système d'information.

2.3.1 Base de données

Définition 2.3.1. Une base de données est une collection organisée d'informations structurées, généralement stockées électroniquement dans un système informatique. Généralement contrôlée par un SGBD (Système de Gestion de Base de Données) (Oracle, 2023).

Définition 2.3.2. Une base de données ou BDD est une collection d'informations organisées

afin d'être facilement consultables, gérables et mises à jour. Les bases de données informatiques sont utilisées dans un grand nombre d'entreprises pour stocker, organiser et analyser les données (LeBigData, 2023).

Pour un SRI, cette base de données est structuré par un expert de base de données (Database Manager). Il est responsable de stockage des documents, des index, ainsi que met en place la relation entre les documents et les index (Fichier inversé). Cette base de données stocke alors l'ensemble de documents utilisé par le système afin de répondre aux besoins de l'utilisateur (Abu-Salih, 2018).

2.3.2 Algorithme de classement

Un algorithme de classement permet de classer les documents suivant l'ordre de pertinence par rapport a la requête de l'utilisateur. Cette algorithme se charge de calculer la pertinence entre la requête et les documents dans la base de données. L'algorithme de classement souvent utilisé dans le modèle vectoriel est déjà présenté dans la Section 1.10.

2.4 Catégorie de recherche

Tout d'abord, il y a quatre catégories de recherche (Yang, 2000) tel que :

— **Recherche simple** : utilise la requête simple
— **Recherche personnalisée** : utilise la requête personnalisée
— **Recherche de dossier** : recherche dans des dossiers
— **Recherche des nouvelles courantes** : utilise la requête des news
— **Contenu web** : suppression des liens mortes

2.5 Type de moteur de recherche

Les moteurs de recherche sont catégorisé en deux catégories : moteur de recherche classique qui travaille avec les contenues statique des documents et le moteur de recherche sémantique (Bellaouar, Djoudi et Zidat, 2009 ; ZIANI, 2022).

Le Tableau 2.1 illustre les différences entre un moteur de recherche classique et moderne.

2.5.1 Classique ou traditionnel

Un moteur de recherche classique ou traditionnel traite les documents dont le contenu est statique c'est à dire, le contenu des documents ne change pas au fil du temps. Ce qui implique, que l'étape d'indexation des documents se fait une fois, il n'y a probablement pas de réindexation des documents. Cette approche est souvent utilisé pour la recherche des livres électronique, etc.

Moteur de recherche classique	**Moteur de recherche moderne**
Nombre de documents inférieur	Nombre de documents supérieur
Nombre d'utilisateurs simultanées inférieur	Nombre d'utilisateurs simultanées supérieur
Type d'informations simple	Multiple type d'informations
Simple, indexation et pondération rapide (Temps, Performance)	Indexation/Clustering/Détermination des pages valués est une challenge

TABLE 2.1 – Catégorie des SRI (YANG, 2000)

2.5.2 Sémantique ou moderne

Ce moteur de recherche provient du traditionnel. Le contenu des documents traités par ce type de SRI sont dynamiques et peuvent varier au fil du temps. Comme par exemple un site WEB qui met a jour fréquemment ses contenus, les suppriment. Dans ce type de moteur de recherche, l'étape d'indexation est de plus en plus complexe, et l'étape de réindexation est nécessaire et souvent periodique pour récupérer les mise a jours des documents. Ce type de moteur de recherche est généralement utilisé pour la recherche sur le WEB.

2.6 Modèle adopté par un SRI

Un Système de Recherche d'Information peut adopter un modèle qui se base sur ces trois modèles qui est le modèle *full-text*, le modèle *keyword-based* et le modèle *hypertext* (BAEZA-YATES et RIBEIRO-NETO, 1999).

— **Full-text** : ce modèle utilise tous les termes de documents comme un terme d'indexation. C'est à dire pas de discrimination des mots vides. L'indexation et la tache de recherche sont simple, mais peut apporter des bruits.
— **Keyword-based** : ce modèle utilise les mots clés pour représenter les documents, avec la suppression des mots vides. Les termes d'indexation sont donc des mots clés.
— **Hypertext** : ce modèle se base sur des liens, souvent utilisé sur les documents WEB.

2.7 Évaluation d'un SRI

Il est important d'évaluer un SRI avant d'implémentation pour l'utilisation finale. Cette étape d'évaluation dépend du but finale du SRI. En général, il y a trois catégories d'évaluation d'un SRI tel que l'*évaluation fonctionnel* qui fait l'analyse des fonctionnalités du système, l'*évaluation de performance* et l'*évaluation de performance de recherche* (BAEZA-YATES et RIBEIRO-NETO, 1999).

Dans l'*évaluation de performance de recherche* troisième catégorie, on utilise des mesures mathématique dont les plus courant sont la *Précision (Precision)* et le *Rappel (Recall)*. C'est souvent dans ce catégorie qui a le plus des difficultés et des défis.

2.7.1 Évaluation fonctionnel

L'évaluation fonctionnel est la première étape d'évaluation d'un SRI. Il met en œuvre l'analyse des deux facteurs suivantes :

— **Analyse de fonctionnalités** : analyser tous les fonctionnalités du système une par une pour voir s'il y a bien des failles ou bug. Cette analyse assurera que l'utilisateur finale du système ne tombe pas sur des problèmes de fonctionnalités ou bug (analyse de l'interface graphique si c'est conviviale, l'accessibilité, etc.).

— **Analyse d'erreur** : pour l'analyse d'erreur, on cherche a trouver un moyen pour faire échouer le système, trouver par tous les moyens de faire des traitements pour que le système fait un erreur. Cette analyse permet de voir des erreurs qui ne sont pas identifié avant et de pouvoir les corriger après.

2.7.2 Évaluation de performance

L'évaluation de performance permet d'identifier la performance et la robustesse d'un SRI. Cette évaluation analyse généralement deux facteurs :

— **Analyse de temps de réponse** : analyser le temps de réponse du système pour répondre a une requête de l'utilisateur. Le but est de minimiser ce temps de réponse pour satisfaire l'utilisateur. Le système doit être rapide en terme de réponse. Avec les machines de plus en plus performants, cet analyse est de moins en moins problématique mais a ne pas ignorer.

— **Analyse de l'espace utilisé** : analyser aussi l'espace disque utiliser par le stockage des index et la collection des documents. Le système doit utiliser le moins d'espace possible, et doit être le plus léger possible.

Dans cette évaluation, on analyse aussi la *performance d'indexation* du SRI, l'*interaction avec le Système d'Exploitation*, ainsi que le *délais dans le canaux de communication.*

2.7.3 Évaluation de performance de recherche

L'évaluation de performance de recherche possède un grand défi, est qui est généralement traité par des chercheurs. Le but est de maximiser la précision des résultats de recherche par rapport au requête de l'utilisateur. Pour bien assimiler cette partie, on va illustré la structure de la collection des documents du point de vue du système, qu'on appelle matrice de contingence dans la Figure 2.1.

Dans ce matrice, il y a quatre catégories tel que :

— **Récupéré et non pertinents** : se sont des documents non pertinents mais jugés pertinent par le système. On cherche a minimiser le nombre des documents dans cette catégorie. On les appelle *faux positif.*

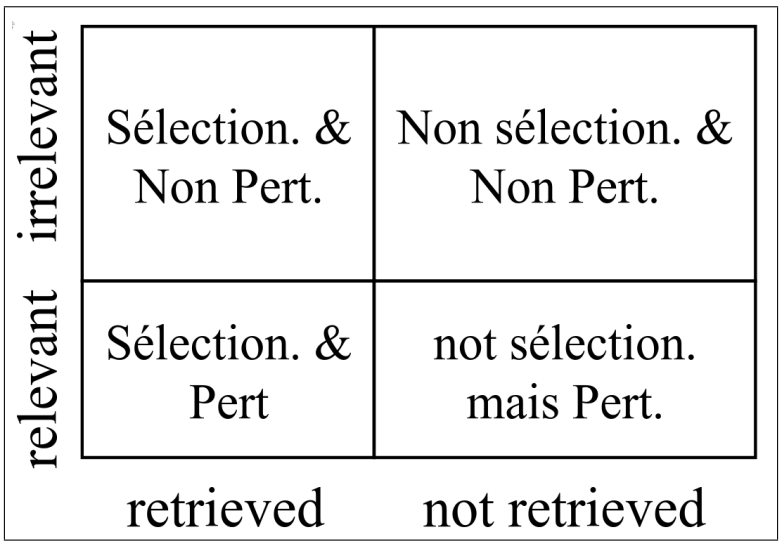

FIGURE 2.1 – Matrice de contingence (ABU-SALIH, 2018)

— **Récupéré et pertinents** : se sont des documents pertinents et qui sont sélectionnés par le système. On cherche a maximiser le nombre des documents dans cette catégorie. On l'appelle *vrai positif*.
— **Non récupéré et non pertinents** : se sont des documents non pertinents qui ne sont pas sélectionnés par le système. On cherche aussi a maximiser le nombre des documents dans cette catégorie. On l'appelle *vrai négatif*.
— **Non récupéré et pertinents** : se sont des documents pertinents mais qui ne sont pas récupérer par le système. On cherche a minimiser le nombre des documents dans cette catégorie. On l'appelle *faux négatif*.

Dans cette évaluation, on va voir quelques mesures parmi les plus populaires, tel que la *précision*, le *rappel*, et la *F-mesure*. A bien noter que la notion d'ensemble des documents pertinents dans cette évaluation est défini par un expert, qui juge qu'un document est pertinent par rapport a un requête. C'est grâce au connaissance de documents pertinents a l'avance qu'on peut juger le système.

On va illustrer dans la Figure 2.2 la classe des documents pour mieux élaborer la notion de *précision* et *rappel*.

2.7.3.1 Précision

Le précision (precision en anglais) est le rapport du nombre de documents pertinent et le nombre de documents retournés par le système. En d'autre terme, la fraction des documents retournés qui sont pertinents. La précision calcule alors la performance du système a récupérer seulement les documents pertinents (ABU-SALIH, 2018 ; BLAIR et MARON, 1985 ; YANG, 2000). Le but est de maximiser cette valeur.

$$Prcision = \frac{Nombre\ de\ documents\ pertinents}{Nombre\ de\ documents\ retournés\ par\ le\ système} = \frac{|Ra|}{|A|}$$

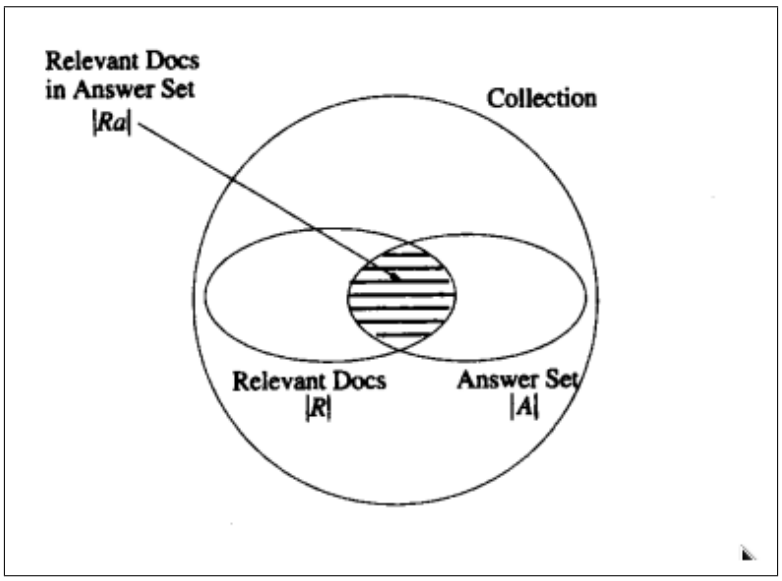

FIGURE 2.2 – Ensemble des documents (BAEZA-YATES et RIBEIRO-NETO, 1999)

2.7.3.2 Rappel

Le rappel (recall en anglais) est le rapport entre le nombre de documents pertinents retournés et le nombre total des documents pertinents. En d'autre terme, c'est la fraction des documents pertinents qui sont retournés par le système. Le rappel permet de savoir comment le système a bien récupérer tous les documents pertinents (ABU-SALIH, 2018 ; BLAIR et MARON, 1985 ; YANG, 2000). Le but est de minimiser cette valeur.

$$Rappel = \frac{Nombre\ de\ documents\ pertinents}{Nombre\ de\ tous\ les\ documents\ pertinents} = \frac{|Ra|}{|R|}$$

Il est nécessaire de noter que le rappel n'est efficace dans le cas où la base de données devient large, car le rappel diminue si la base de données augmente.

2.7.4 Exemple de précision et rappel

Un exemple de calcul de précision et rappel est illustré dans la Figure 2.3.

Et la courbe (relation) entre la précision et le rappel est illustré dans la Figure 2.4

2.7.5 Problème de précision et le rappel

Citons quelques problèmes de la précision et du rappel, comme l'estimation correcte du rappel maximum pour une requête nécessite une connaissance détaillé de tous les documents dans la collection, la précision et le rappel sont deux approche proche qui prend différents aspects de l'ensemble des documents retournés, ainsi ils mesurent l'efficacité a travers une requête traité en groupe ce qui n'est pas adaptable pour un SRI moderne (BAEZA-YATES et RIBEIRO-NETO, 1999).

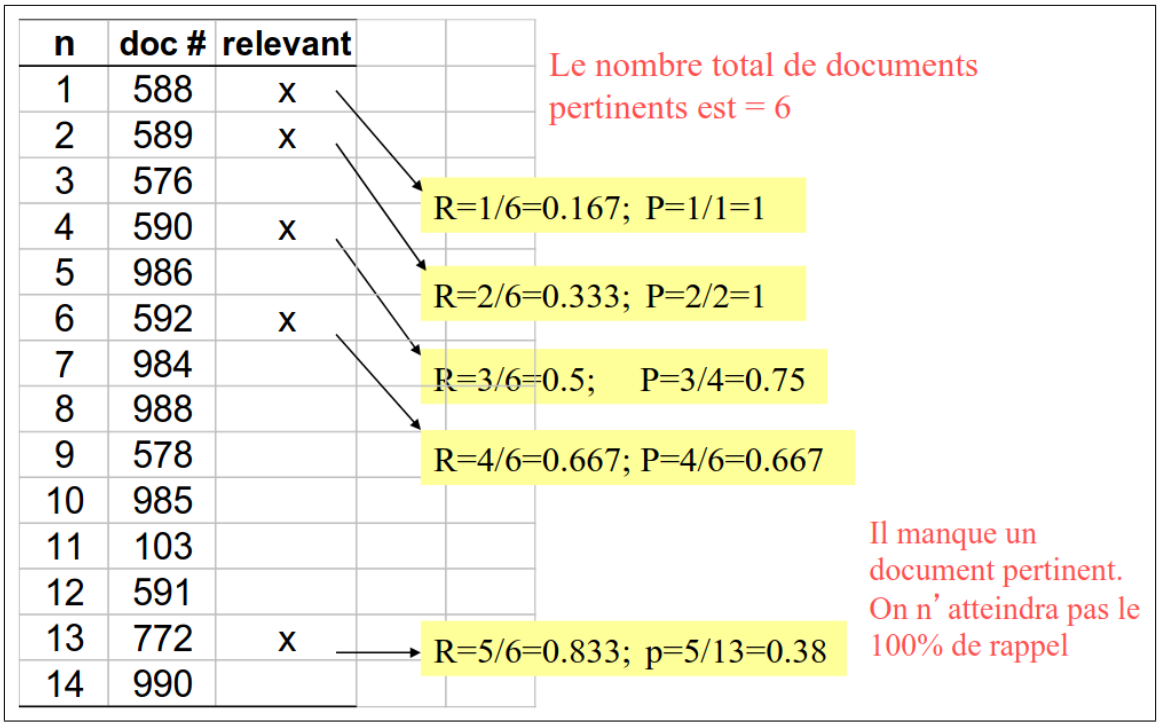

FIGURE 2.3 – Exemple de calcule de précision et rappel (BOUGHANEM, s. d.)

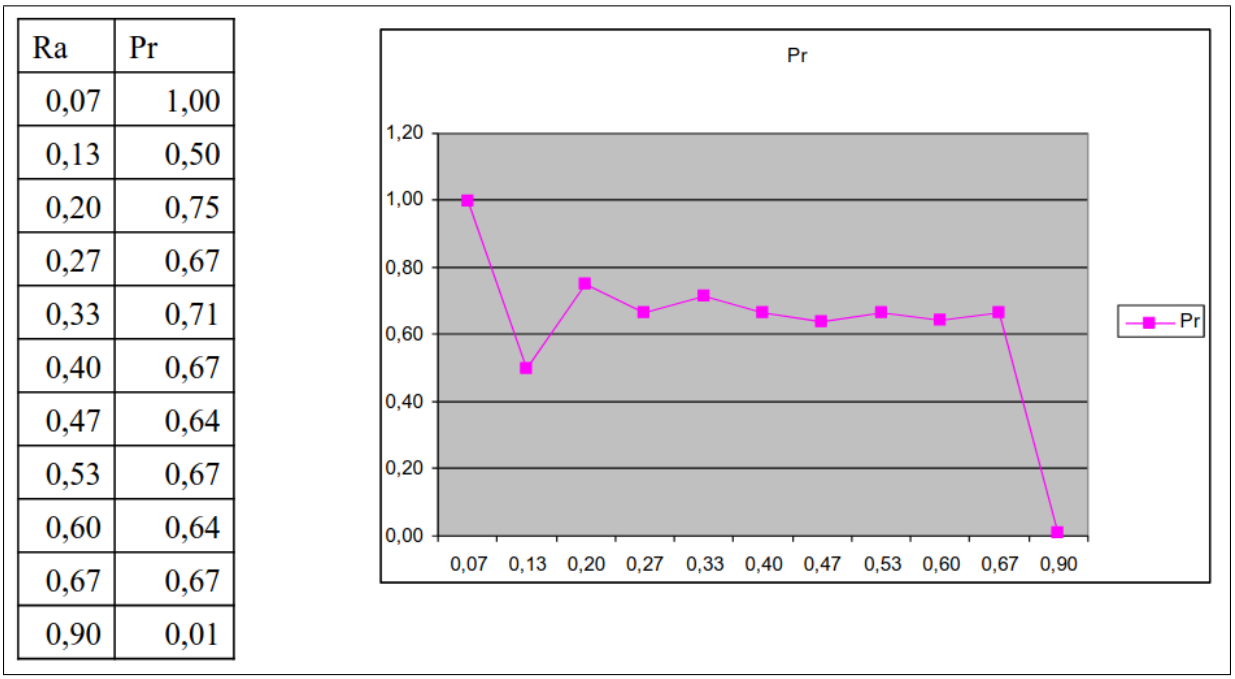

FIGURE 2.4 – Courbe précision-rappel (BOUGHANEM, s. d.)

2.7.6 Moyenne harmonique

Cette mesure utilise la précision et le rappel.

$$F(j) = \frac{2}{\frac{1}{R(j)} + \frac{1}{P(j)}}$$

Où :
R (j) est le rappel pour le j-ième document.
P (j) est la précision pour le j-ième document.
F (j) est la moyenne harmonique

2.7.7 Et d'autres mesures

D'autres mesure de performance sont utilisés comme *accuracy*, *F-measure*, *E-measure*, *x-precision*, etc. Il y a aussi de notion de silence et bruit (Baeza-Yates et Ribeiro-Neto, 1999 ; Mazari, 2022).

2.8 Analyse du moteur de recherche existant

On peut trouver des thèses malagasy dans différentes moteur de recherche qui existent, mais dans le cadre de ce mémoire, on va analyser quelques uns qui sont populaire (dans le cadre des thèses malagasy) et qui concerne le mieux notre étude.

2.8.1 Thèses malgache en ligne

Thèse malgache en ligne (Université d'Antananarivo, 2016) est un SRI orienté WEB spécialisé pour la recherche des thèses soutenues dans les six universités publiques de Madagascar depuis 2006. Actuellement il regroupe actuellement **31 160** documents, plus précisément des thèses. Ce SRI est hébergé dans le serveur de l'université d'Antananarivo, et est mis en place par les équipes au sein de l'université. Le statistique des documents est dans la Figure 2.5, les résultats de recherche ainsi que le système de pagination est illustré dans la Figure 2.6 et la Figure 2.7.

Ce SRI a pour objectif de faciliter les recherches de thèses, généralement d'auteur malagasy afin d'explorer et de mettre en évidence les fruits de recherche malagasy. Ainsi de faciliter l'accès a ces documents pour les enseignants ainsi que pour les étudiants qui cherche des travaux rattachés a son thème.

L'avantage majeur de ce système est son rapidité en terme de réponse a une requête. Et il est possible de faire une recherche par un titre, auteur et contenu, ainsi que la mise en place de système de pagination pour classer les documents plus pertinents dans la première page,

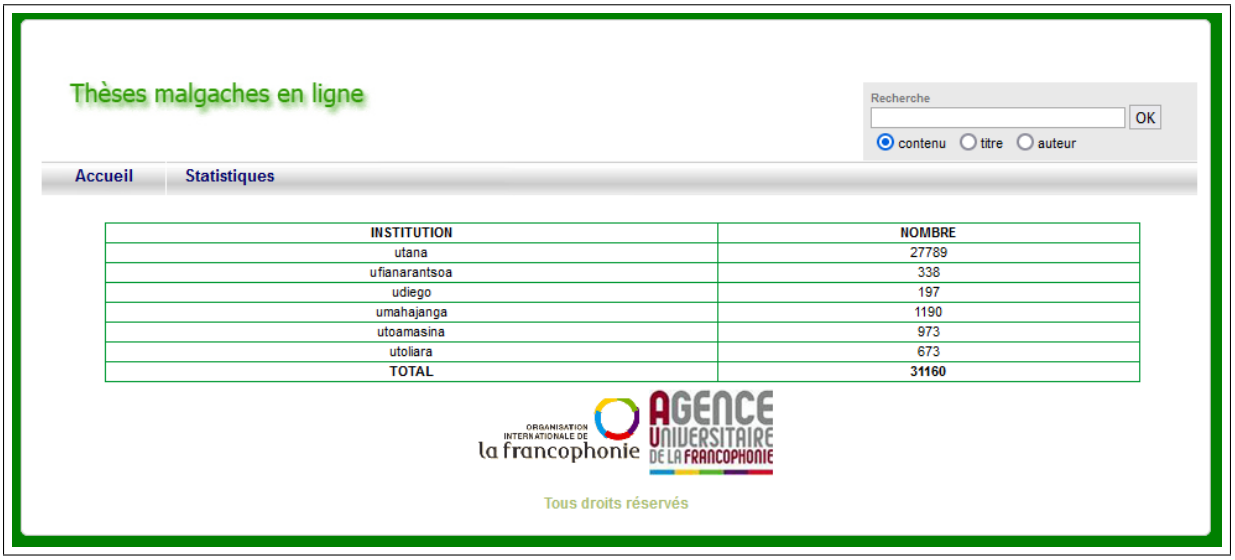

FIGURE 2.5 – Statistique des documents (UNIVERSITÉ D'ANTANANARIVO, 2016)

et de naviguer dans les autres pages. Ce SRI classe les résultats suivant l'ordre de pertinence décroissant. L'interface utilisateur est très simple ce qui implique que le chargement de la page est rapide et facile a prendre en main.

Par contre, ce système a quelques lacunes malgré ces avantages. La première problème est l'insuffisance des documents stockés. Vu le nombre d'étudiant ayant soutenu dans les six universités publiques de Madagascar, ce nombre n'est pas suffisant. La deuxième, les documents sont limités pour les établissements publiques, alors qu'il pourra être intéressant d'inclure les documents des établissement privés. La troisième c'est qu'il y a pas de système de catégorisation des documents. Et d'autres problèmes comme le nombre de documents retournés sont nombreux qui implique souvent beaucoup de pagination des résultats (pas de limite de pertinence). Et aussi, il est impossible de voir la résumé d'un document particulier ou afficher une aperçu du document. Et finalement il y a pas de notion de sécurité de document c'est a dire tous les documents sont exposés publiquement en tant qu'utilisateur anonyme.

2.8.2 Bibliothèques universitaire en ligne

Les bibliothèques universitaire en ligne n'est plus ni moins qu'une version numérique des bibliothèques. Chaque université publique de Madagascar possède un. En général, c'est pas un SRI complet mais juste une liste des documents disponibles avec un simple recherche généralement sur le titre ou auteur.

L'objectif est d'exposer des documents pour faciliter l'accès aux étudiants, enseignants ou des personnes voulant accéder a des ressources. Mais en général, ce système fonctionne très bien avec un recherche simple dans le cas des livres par exemple, mas s'avère compliqué pour des recherches avancés. Ainsi les documents sont incomplètes.

Dans le cadre de recherche des thèses malagasy, ce système est moins efficace que *Thèse*

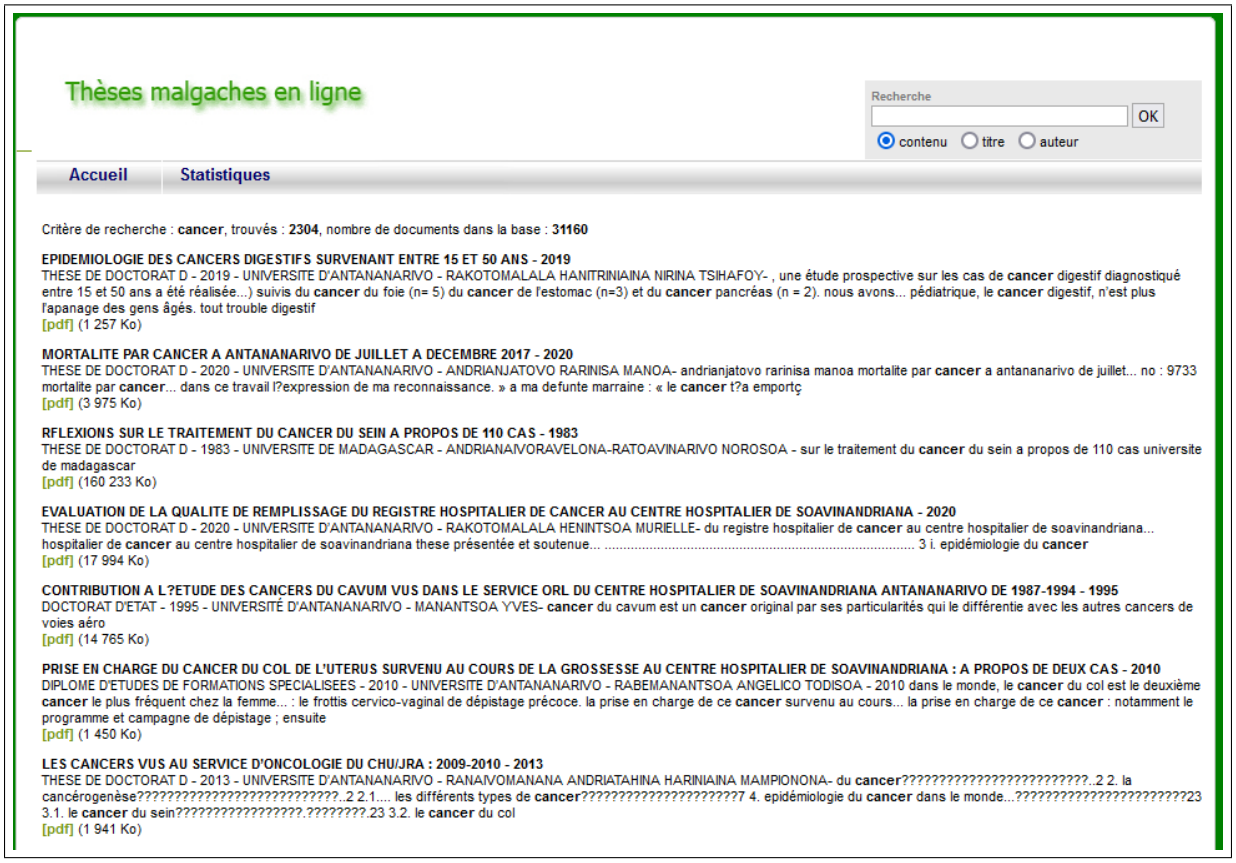

FIGURE 2.6 – Résultat de recherche (UNIVERSITÉ D'ANTANANARIVO, 2016)

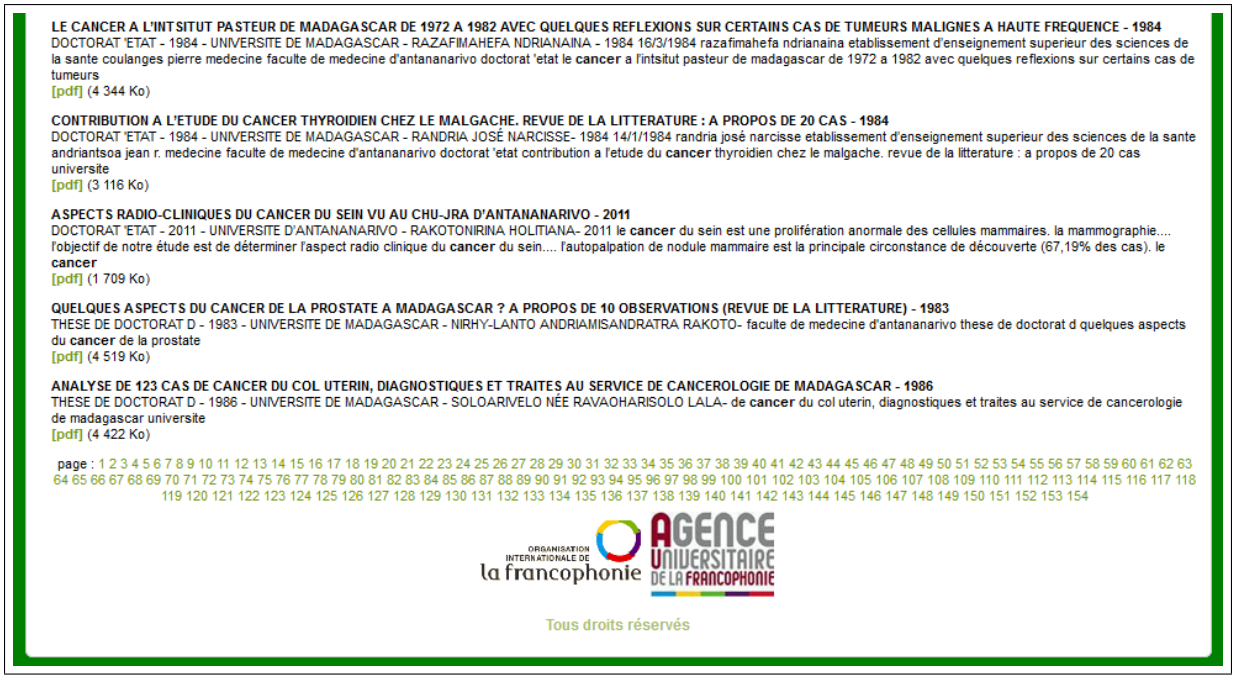

FIGURE 2.7 – Système de pagination (UNIVERSITÉ D'ANTANANARIVO, 2016)

malgache en ligne.

2.8.3 Google Scholar, theses.fr et d'autres

Les géants du moteurs de recherche académique tel que *google scholar*, *Mémoire online*, *HAL*, et d'autres peuvent bien être utilisé pour rechercher des thèses malagasy. Ces systèmes sont performants, puissant en terme d'algorithme utilisés, et stock une grande volume des documents. Certains système propose d'aperçu du document, exportation des bibliographie, liste des articles connexes ainsi qu'un système de sécurité et d'authentification. Il est aussi possible de faire une filtre par année ou par catégorie.

Malgré tous ces atouts, c'est pas suffisant pour les travaux de recherches malagasy. Ces moteurs n'arrivent pas a indexer beaucoup des documents (thèses) malagasy. Par exemple, dans thèses.fr, il y a seulement 41 thèses en cotutelle pour l'université d'Antananarivo (THESES.FR, 2023). Alors pour rechercher des travaux de recherche malagasy dans ces moteurs est un véritable défi.

2.9 Conclusion

En résumé, un Système de Recherche d'Information ou Moteur de Recherche est un système actuellement indissociable de la vie quotidienne, académique, personnel ou même professionnel. Il a pour but de satisfaire le besoin d'information de l'utilisateur en utilisant des algorithmes et des traitements particulières sur les documents et la requêtes. Un SRI doit passer certains test avant d'être finalement implémenté pour l'utilisation finale. Ces testes sera réalisé pas des experts du domaine, et en utilisant des corpus de test (collection de test) pour pouvoir déterminer la pertinence et la performance du SRI. Il y a actuellement beaucoup des moteurs de recherche, certains sont spécialisés dans un domaine spécifique (news, sports, politiques, etc.), mais on a analysé justement quelques un qui est important dans le cadre de ce devoir.

Chapitre 3

Traitement de Langage Naturel (TLN)

3.1 Introduction

Le Traitement de Langage Naturel (Natural Langage Processing) est une approche pour analyser des textes qui est basé sur les théories et la technologies. C'est un domaine très répandu actuellement dans la recherche et développement (Liddy, 2001). La NLP est utilisé dans la *recherche d'information*, *traduction des textes*, *traduction des voix*, *système de question réponse* ainsi que diverses domaines. Citons quelques définitions parmi tant d'autres.

Définition 3.1.1. Le Traitement de Langage Naturel (TLN) est un discipline qui combine la linguistique, l'informatique et l'intelligence artificielle pour analyser et étudier les interactions entre uns système informatique et une langage naturel humain. La Recherche d'Information est parmi l'une de tâche du NLP (Andrea Ferrario, 2020).

Définition 3.1.2. Selon Liddy, le TLN est un ensemble des techniques informatiques théoriques, pour analyser et représenter naturellement des textes (dans un langage quelconque) en passant par un ou plusieurs niveaux d'analyse linguistique dans le but de parvenir a traiter le langage humain dans une tâche ou applications (Liddy, 2001).

L'analyse de langage, porte a connaître la structure de langage tel que les mots, les significations, combinaison des mots, ainsi que la contribution des mots au sens de la phrase. Et aussi d'analyser le fonctionnement du monde et le raisonnement de l'humanité dans le monde (Tannier, 2006).

Le domaine du TLN comporte des divisions tel que :

— **Traitement de Langage Naturel (NLP)** : production de représentation significatif. Équivalent du rôle de lecteur et écouteur.
— **Génération de Langage Naturel (NLG)** : production de langage a partir d'une représentation. Équivalent du rôle d'auteur et interlocuteur.
— **Compréhension de langage (Language Understanding)** : Termine avec un langage

orale.

— **Compréhension de langage orale (Speech Understanding)** : Démarre avec un langage orale. Permet de savoir comment un langage orale va se traduire en texte.

Le domaine de TLN par rapport a l'intelligence artificielle est illustré dans la Figure 3.1.

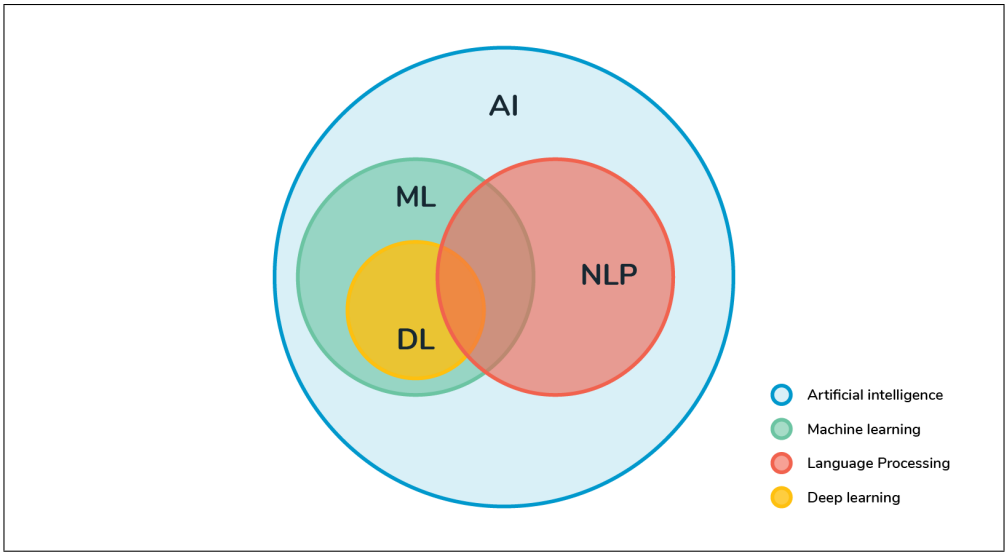

FIGURE 3.1 – Intelligence Artificiel et Traitement de Langage Naturel (DEVOPEDIA, 2023)

Certains notion qu'on abordera ici sont déjà présenté dans la Section 1.8, et qui ne seront pas détaillés.

3.2 Un peu d'histoire

Cette brève histoire est tiré de l'ouvrage de LIDDY (LIDDY, 2001). La recherche en Traitement de Langage Naturel trouve ses origines dans les années 1940. Le *MT (Machine Translation)* est la première application en TLN qui est une application d'ordinateur. En 1946, Weaver et Booth démarre l'une de projet récent de MT, encore sur la traduction basé sur l'expertise de voler le code d'ennemie durant la deuxième guerre mondiale. Ce projet a suggérer l'utilisation de la cryptographie et de la théorie de l'information pour le traduction de langage. Puis la recherche sont devenu varié dans les institutions en États-Unis d'Amérique (USA).

En 1957, Chomsky a publié *Syntactic Structure*, qui introduit l'idée de grammaire générative pour remédier aux problèmes rencontré dans les années antérieurs tel que l'ambiguïté syntaxique, ... Une autre domaine de TLN commence aussi a émerger, comme la reconnaissance vocale.

La communauté de *traitement de langage* et la *communauté vocale* (Speech community) étions divisé alors en deux camps : d'une part la communauté de traitement de langage dominé par la théorie perspective de grammaire germinative et hostile en méthode statistique ; et d'autre part la communauté vocal (Speech Community) dominé par la théorie statistique de l'information et hostile en théorie linguistique.

En 1950, l'homme pense que la meilleur qualité totalement automatique peut produire des résultats indiscernable de la translation humaine, et qu'un tel système peut être opérationnel dans quelques années.

Enfin, le TLN a connu beaucoup d'évolution ainsi des nombreuses lacunes. A partir des années 1980, le TLN a évolué beaucoup plus vite par la dispositions des ressources computationnel ainsi que des ordinateurs performants.

3.3 Les niveaux de langage

Un langage comporte généralement cinq niveaux ou étape de traitement (Dale, Robert et Moisl, Hermann et Somers, Harold, s. d. ; Tannier, 2006), illustré dans la Figure 3.2 :

— **Phonétique et phonologie** : analyse la liaison des mots et des phrases au son qui les réalisent a l'oral. Cette niveau n'est pas utile dans le traitement de langage textuelle.
— **Morphologie** : analyse la façon dont les mots sont construits, et de connaître leurs rôles dans la phrase.
— **Syntaxe** : analyse la façon dont les mots se combinent pour former des *syntagmes*, puis des proposition et enfin des phrases correctes. Un **syntagmes** est un unité syntaxique intermédiaire entre le mot et la phrase, et comprenant un seul noyau et souvent des compléments.
— **Sémantique** : analyse la façon dont les mots font du sens lorsqu'ils sont insérés dans une phrase indépendamment du contexte.
— **Pragmatique** : analyse la façon d'interpréter les phrases selon leur contexte d'énonciation comme l'interlocuteur, phrase précédente, connaissance commune du monde, ...

3.4 Phonologie

La phonologie interprète le langage parlé (son de parole) dans un mot (Liddy, 2001). Il y a trois règles tel que :

— **Règle phonétique** : pour les sons dans les mots.
— **Règle phonémique** : pour les variations de prononciation quand les mots son prononcé ensemble.
— **Règle prosodique** : pour la variation d'un stresse et d'intonation dans un phrase.

3.5 Morphologie

La morphologie est l'étude de la forme des mots tel que leur *flexion* : indication de cas, genre, nombre, mode et temps ; leur *dérivation* : préfixes, suffixes et infixes ; leur *composition* : mots composés. C'est aussi une analyse morphosyntaxique, qui analyse des règles de combinaison des morphèmes selon la configuration syntaxique de l'énoncé. La morphologie consiste

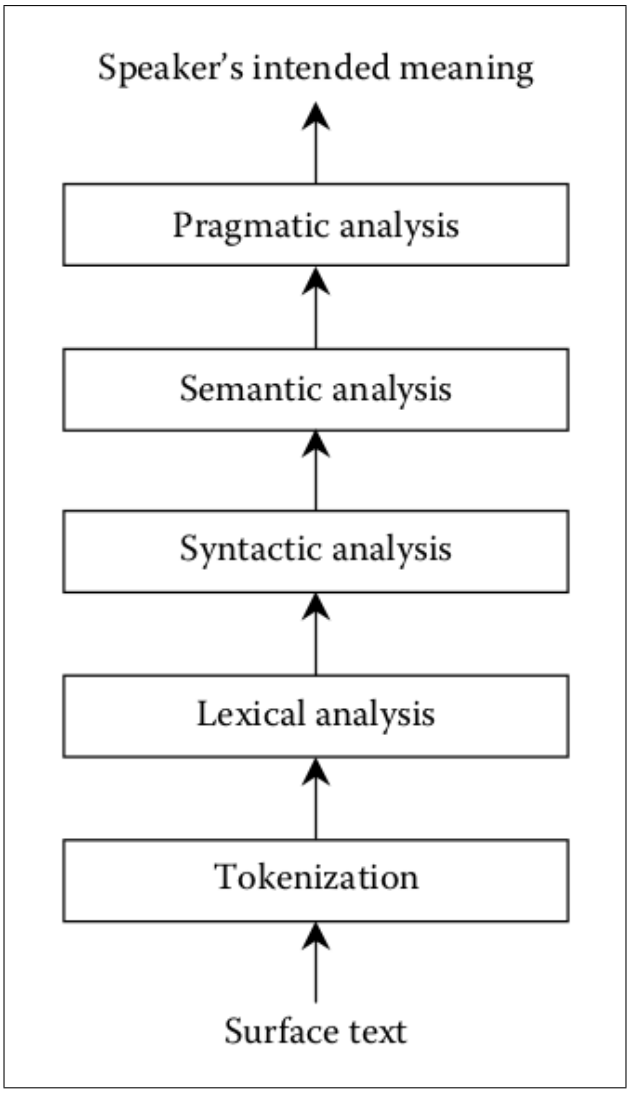

FIGURE 3.2 – Étape d'analyse de texte dans le TLN (DALE, ROBERT ET MOISL, HERMANN ET SOMERS, HAROLD, s. d.)

a segmenter des textes en unités élémentaire qu'on appelle *tokenisation* et à déterminer les différentes caractéristiques de ces unités (LIDDY, 2001 ; TANNIER, 2006).

Un humain est capable de faire ce traitement pour comprendre le sens, vu que chaque sens des morphèmes ne change pas dans tous les mots, et même pour un système TLN qui consiste a reconnaître le sens porté par chaque morphème pour représenter le sens (LIDDY, 2001).

Définition 3.5.1 (Lemme)**.** On appelle *lemme* la racine d'un mot, sans ses marques d'accord, de conjugaison, de cas. L'entrée d'un dictionnaire est généralement un lemme. Le lemme est nécessaire dans toute analyse sémantique. (TANNIER, 2006)

Définition 3.5.2 (Flexion)**.** Les *flexions* sont les modifications opérées sur le lemme pour distinguer les formes de conjugaison (personne, temps, mode, voix – flexion verbale) ou le genre, le nombre et le cas (flexion nominale). L'opération qui consiste à retrouver ces informations est la *lemmatisation*, ou souvent appelé *racinisation (stemming)* (TANNIER, 2006).

Définition 3.5.3 (Morphème)**.** On appelle *morphème* le plus petit unité ou unité minimale de sens (LIDDY, 2001). Par exemple, le mot prétraitement est composé de trois morphèmes tel que le préfixe **pre**, la racine **traite** et la suffixe **ment**.

Cette étape est une partie essentiel du TLN, qui est nécessaire pour définir les caractères, mots et phrases dans un document. Elle relève un défi tel que la *résolution des ambiguïtés* dans le langage naturel ainsi que de convertir des fichiers textes dans un séquence de texte bien défini avec un sens. Cette étape peut être divisé en deux parties : *Triage de document (Document Triage)* et *Segmentation de texte (Text Segmentation)* (DALE, ROBERT ET MOISL, HERMANN ET SOMERS, HAROLD, s. d.).

3.5.1 Triage de document

Le triage de document c'est l'action de convertir un ensemble des fichiers en documents textuelles bien définie. Il se décompose en trois étapes tel que :

1. **Identification d'encodage de caractère (Character Encoding Identificator)** : permet d'identifier l'encodage de caractère utilisé dans le fichier a convertir, par exemple si le fichier utilise l'encodage UTF-8, etc.
2. **Identification de langage (Language Identification)** : permet de déterminer dans quelle langue le fichier est écrit (Français, Anglais, etc.).
3. **Sectionnement de texte (Text Sectionning)** : permet d'identifier le contenu textuel du fichier pour pouvoir le convertir en document textuel bien défini.

3.5.2 Segmentation de texte

La segmentation de texte est un étape cruciale et complexe dans le Traitement de Langage Naturel. Il consiste a convertir un document textuel bien définie en composant des mots et

des phrases (Dale, Robert et Moisl, Hermann et Somers, Harold, s. d.). C'est aussi la transformation d'un alignement d'un caractère en une unité élémentaire souvent des mots et phrase (Tannier, 2006). Généralement, il y a trois étapes tel que la *segmentation de mot (Word segmentation)*, la *normalisation de texte (Text Normalization)* et la *segmentation de phrase (Sentence Segmentation)*.

Pour la segmentation de texte, il faut définir une liste de caractères délimiteur pour pouvoir segmenter le texte (espace, ponctuation, apostrophe). Pour les langages dont l'espace n'est pas un délimiteur comme la langue chinoise par exemple, cette segmentation est plus complexe (Dale, Robert et Moisl, Hermann et Somers, Harold, s. d.). Et même avec la langue qui a comme délimiteur les trois caractères cités ci-dessus, il est quand même difficile de séparé les mots, car l'apostrophe peut être utilisé dans un seul mot comme *aujourd'hui*, aussi un unité élémentaire peut contenir des espaces comme *pomme de terre* (Tannier, 2006).

En pratique le système de segmentation du texte utilise une liste de séparateurs par défaut, a laquelle ils ajoutent des connaissances lexicales et morphosyntaxiques pour traiter les cas ambigus. Chaque lange possède ses propres connaissances lexicales.

Segmentation de mot

La segmentation de mot c'est l'action couper un séquence des caractères en texte par la localisation de délimiteur de mot (*Word boundaries*) pour pouvoir être utilisé dans la traitement linguistique. Un délimiteur permet de définir le point où un mot est terminé et qu'un autre commence. Les mots obtenues sont appelées des *tokens*, et la façon d'obtenir ces tokens est la *tokenisation* ou *tokenization* en anglais (Dale, Robert et Moisl, Hermann et Somers, Harold, s. d.).

Pour les langages délimité par un espace (*space-delimited*), souvent des langages européennes, les mots sont séparés par des espaces. Tandis que pour des langages qui ne sont pas délimités par un espace comme le Chinois et le Thaï, il n'y a pas d'indication sur le délimiteur (Dale, Robert et Moisl, Hermann et Somers, Harold, s. d.). Dans le cadre de ce devoir, on travaillera sur des langages délimité par un espace.

Dans un langage délimité par un espace, les ponctuations sont souvent traité comme des tokens séparés, mais qui varie d'un langage a un autre. Mais pose souvent des problèmes comme des abréviation, les quottes et les apostrophes. (Dale, Robert et Moisl, Hermann et Somers, Harold, s. d.)

Normalisation de texte

La normalisation de texte permet de fusionner les différentes formes de token en une forme canonique normalisé par exemple Mr et Monsieur. Cette normalisation permet aussi de normaliser des dates, des heures, des formats monétaires et d'autre informations numériques. Par exemple, les trois tokens écrit $300 peut être prononcé comme « trois cent dollars » et la norma-

lisation de texte peut convertir le token original en token désiré. (Dale, Robert et Moisl, Hermann et Somers, Harold, s. d.)

Segmentation de phrase

C'est de déterminer comment le texte doit être divisé en phrase pour un traitement plus avancé. En identifiant les délimiteurs de phrase (souvent un point pour la langue délimité par un espace). Souvent la segmentation de phrase est référencé comme *détection de délimiteur de phrase* ou *désambiguïsation de délimiteur de phrase* ou encore *reconnaissance de délimiteur de phrase* (Dale, Robert et Moisl, Hermann et Somers, Harold, s. d.).

3.5.3 Catégories grammaticales

Les catégories grammaticales est une information morphosyntaxique importante. Ces catégories sont souvent des noms, verbe, adjectif, mais qui varie selon le système. Par exemple *Penn Treebank* contient 45 catégories. Sur la Figure 3.3 un exemple de catégorie grammaticale pour la phrase « *Qui veut noyer son chien l'accuse de la rage* », et qui illustre aussi la problème d'ambiguïté pour cette phrase. En général 25% du lexique est de forme ambigus pour la langue française. Et un exemple de description morphosyntaxique illustré dans la Figure 3.4.

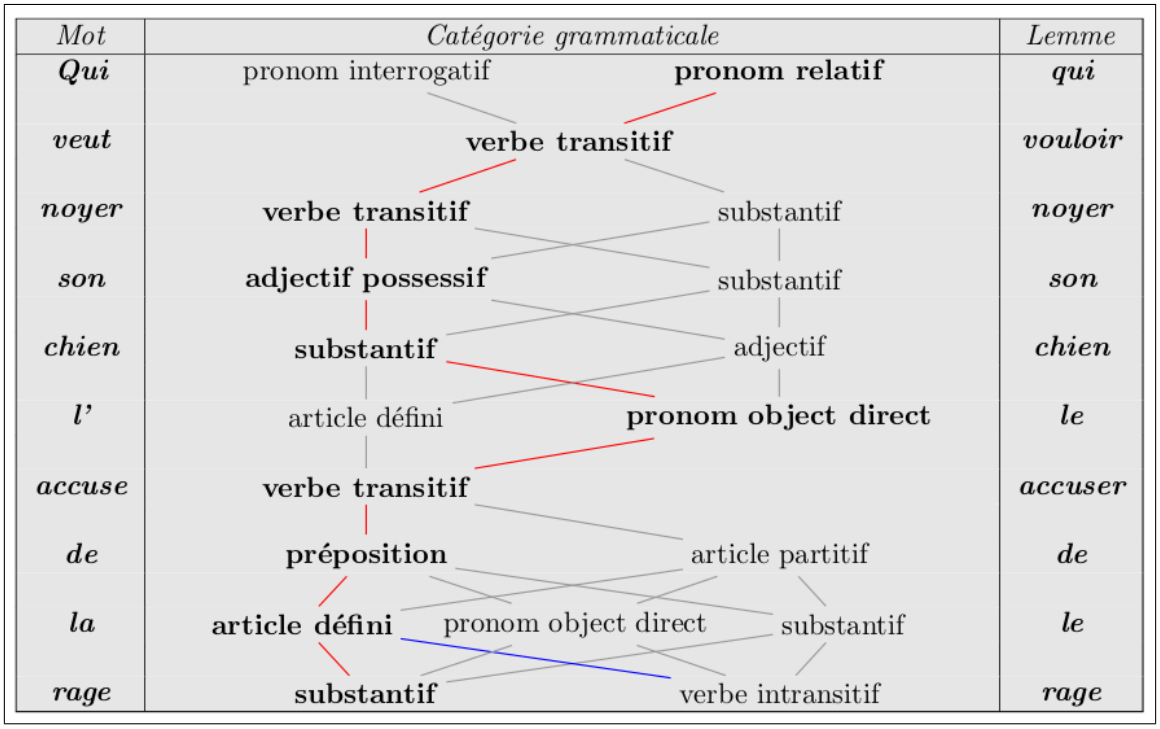

Figure 3.3 – Catégorie grammaticale (Dale, Robert et Moisl, Hermann et Somers, Harold, s. d.)

3.6 Analyse lexicale

Un humain ou un Système TLN interprète le sens de chaque mot individuellement. L'analyse lexicale permet alors de déterminer le sens de chaque mot individuellement. Le but est de

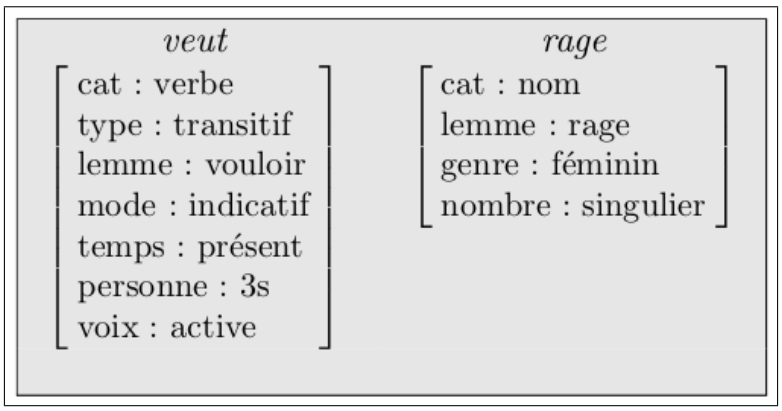

FIGURE 3.4 – Description morphosyntaxique (DALE, ROBERT ET MOISL, HERMANN ET SOMERS, HAROLD, s. d.)

contribuer au compréhension de niveau de mot par une affectation de tag *part-of-speech* pour chaque mot. Un mot qui a un sens possible peut être remplacé par un représentation sémantique de ce sens (LIDDY, 2001).

Un analyse lexicale utilise un connexion, qui est déterminé et choisi par le système TLN. Un lexicon peut être **simple** c'est à dire des mots et ses *pos (part of speech)* ou peut être **complexe** et contient l'information de classe sémantique des mots (LIDDY, 2001).

3.7 Syntaxe

La syntaxe décrit comment les lemmes ou flexions sont ordonnées pour créer des constituants, composant aux-mêmes des phrases. Souvent, l'analyse syntaxique est représenté de façon hiérarchique. Dans l'analyse syntaxique, on travaille avec les mots, les phrases, ainsi que l'étape de la formation du mots a la phrase (TANNIER, 2006). L'analyse syntaxique est une analyse complexe avec différentes niveaux, les détails ne seront pas traité dans ce devoir, mais présent dans « Traitement automatique du langage naturel pour l'extraction et la recherche d'informations », section 3 (TANNIER, 2006).

L'analyse syntaxique selon LIDDY, se focalise sur l'analyse des mots dans une phrase, pour découvrir la structure grammaticale de la phrase, qui nécessite l'utilisation de la grammaire et un parseur. Dans l'analyse syntaxique, l'ordre et le dépendance des mots contribue au sens : par exemple, la phrase *« Le chat attaque le chien »* et la phrase *« Le chien attaque le chat »* ont un sens différent car ses termes de syntaxe sont différents, pourtant ils sont composé des mêmes mots (LIDDY, 2001).

3.7.1 Analyse superficielle

Le but de l'analyse superficielle ou partielle (TANNIER, 2006), est de reconnaître les syntagmes simples, non récursif d'un énoncé sans lier les uns aux autres ; d'obtenir des résultats moins riches mais plus sûr et plus rapide. Cette approche s'oppose a l'analyse profonde ou

complète qui cherche a regrouper chaque phrase dans une unique représentation.

Cass est un exemple de d'analyseur partielle efficace, qui consiste en une cascade d'automates à états finis.

3.7.2 Analyse en dépendance

L'analyse syntaxique en dépendance diffère surtout de l'analyse en constituants (comme les règles hors-contexte) par le mode de représentation. Les deux approches n'ont pas de différence au niveau de leur couverture ou de leur expressivité (A reformuler). L'idée est de relier les mots et non les constituants (TANNIER, 2006).

Un exemple de grammaire de dépendance est la *Link Grammar* qui définit la lexique et les contraintes d'attachement, comme illustré dans la Figure 3.5 et la Figure 3.6

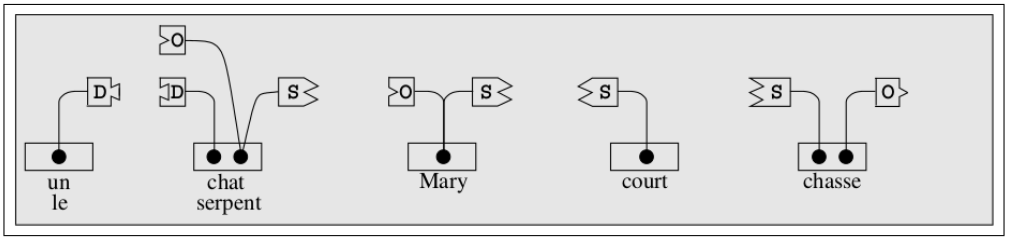

FIGURE 3.5 – Exemple de définition de la lexique *Link Grammar* (TANNIER, 2006)

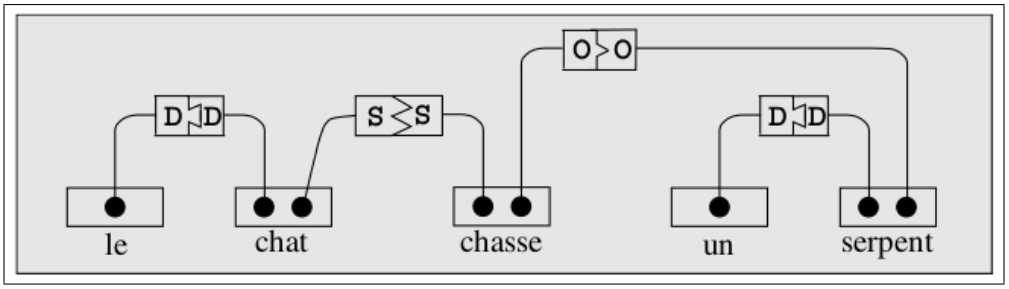

FIGURE 3.6 – Exemple d'analyse syntaxique avec *Link Grammar* (TANNIER, 2006)

3.8 Sémantique

Un approche sémantique peut être utilisé pour réduire les ambiguïtés syntaxiques, a mieux cibler des contextes (en RI par exemple), mais son but finale est de représenter formellement l'information véhiculée par un énoncé et éventuellement d'en inférer des nouvelles connaissances ou une réponse à la question posée dans le cas où l'énoncé est une question (TANNIER, 2006). Il a pour but d'associer a un séquence de mots une représentation interne de son sens, tout en prenant compte l'utilisation futures des résultats obtenues.

Selon beaucoup des personnes, c'est dans le niveau sémantique qu'on détermine la signification, pourtant tous les niveaux contribuent au signification. La sémantique détermine alors la signification possible d'une phrase en se concentrant sur l'interaction parmi les significations

de niveau des mots dans une phrase, avec la possibilité de désambiguïsation sémantique des mots aux multiple sens. Par exemple si on prend le mot **pile**, ça peut être un pile pour alimenter un appareil électronique, ou ça peut être un structure dans un langage de programmation informatique.

Selon TANNIER, il y a quatre méthodes :

— **L'analyse profonde** : permet d'obtenir une représentation complète de l'énoncé
— **Interprétation sémantique grammaticales** : qui s'appuie sur une analyse syntaxique totale
— **Grammaires sémantiques** : qui modélisent les données spécifiques
— **Patrons sémantique** : qui détectent les informations prédéfinies dans un texte

On distingue deux types de sémantique tel que la *sémantique grammaticale* qui se charge a construire un sens de l'énoncé globale, et la *sémantique lexical* qui étudie la participation des mots a ce sens et tous les ambiguïtés qu'ils provoquent.

MAZARI ; ZIANI (MAZARI, 2022 ; ZIANI, 2022) ont chacun développé un moteur de recherche basé sur cette notion de sémantique pour améliorer la qualité de recherche, et analyser les ambiguïtés dans un terme de recherche des utilisateurs.

Quelques relations importantes entre les mots

Citons les relations importantes entre les mots (TANNIER, 2006) qui sont nécessaire en analyse sémantique tel que :

— **Polysémie et l'homonymie** : propriété de certains formes graphiques (signifiants) de renvoyer à plusieurs sens (signifié). Par exemple, le mot *bureau*.
— **Synonymie** : lien entre deux mots ayant la même sens.
— **Hyponymie** : relation d'inclusion entre deux mots dont l'un (hyponymie) est plus spécifique que l'autre (hyperonyme). Par exemple le mot *gorille* est un hyponymie du mot *quadrumane*, le mot *fleur* est un hyperonymie du mot *tulipe*.
— **Méronymie et holonymie** : relation de partie a tout. Par exemple, une serrure est une partie d'une cage (méronymie), un bâtiment contient une pièce (holonymie).

Ces relations existant entre les mots sont répertoriés dans des base informatiques dont le plus utilisés est **WORDNet**.

3.9 Pragmatique

LIDDY cite un niveau supplémentaire entre la sémantique et la pragmatique, c'est le **discours** qui travaille sur un texte plus longue qu'une phrase et interprète le sens en connectant tous les phrases. Il y a deux types de traitement de discours tel que : l'*Anaphore* et le *Discours/text structure recognition*.

Tandis que l'analyse pragmatique sert dans un situation bien spécifique, utilise de contexte qui dépasse le contenu de compréhension de texte (LIDDY, 2001). En d'autres termes, il regroupe un grand nombre de domaine qui englobent tous les problèmes qui ne pouvant être traités avec la syntaxe et la sémantique (TANNIER, 2006). Cette analyse nécessite l'utilisation des connaissances extra-linguistiques sur le contexte et du discours. Pour cette raison que les application pratiques sont rares, pourtant il y a quelques uns (TANNIER, 2006) comme la *déictique*, l'*implicatures conversationnelle* et la *présupposition*.

3.10 Approche du TLN

Il y a généralement trois approche dans la Traitement de Langage Naturel, tel que l'*approche symbolique*, l'*approche statistique* et l'*approche connexionniste*.

3.10.1 Approche symbolique

Cette approche a coexisté avec l'approche statistique le jour après la naissance de ce domaine. Elle fait une analyse profonde des phénomènes linguistiques, et qui est basé sur la représentation explicite des faits a propos d'un langage a travers une connaissance bien compris. Cette approche a été trouvé dans un système basé sur des règles (ensemble des règles, moteur d'inférence et un espace de travail) ; ou logique : structure sans forme de proposition logique.

L'approche symbolique est appliqué dans diverses domaines de recherche comme l'*extraction d'information*, la *catégorisation de texte*, *résolution d'ambiguïté*, et l'*acquisition lexicale*.

3.10.2 Approche statistique

Cette approche, souvent utilisé un large volume de texte (corpora), emploi des techniques mathématiques variés pour développer des modèles approximatif généralisé de phénomène linguistique. Cette approche utilise des données observables comme source primaire d'évidence. Le modèle statistique souvent utilisé est le *HMM* ou *Hidden Markov Model*.

L'approche statistique est appliqué dans la *reconnaissance vocale*, *acquisition lexicale*, *parsing*, *part-of-speech tagging*, *machine de translation statistique*, *collocations*, *apprentissage statistique de grammaire*.

3.10.3 Approche connexionniste

Cette approche est apparu dans les années 1960, est comme l'approche statistique qui développe des modèles généralisés depuis l'exemple de phénomène linguistique. Elle combine l'apprentissage statistique avec différentes *théories de représentation* qui permet la transformation, l'inférence et la manipulation de formule logique.

Certains modèles de l'approche connexionniste s'appelle **modèle localiste** qui peut faire des tâches comme *désambiguïsation de sens de mot*, *génération de langage*, et *inférence limité* ; et **modèle distribué** qui est utilisé pour les tâches comme la *parseur des syntaxes (syntactic parsing)*, *translation dans un domaine limité ou spécifique* et *recherche associative*.

3.11 Application de TLN

La Traitement de Langage Naturel s'applique dans diverses domaines de recherche que ce soit textuelle ou sonores.

La Recherche d'Information (RI)

Dans la recherche d'information, on travaille sur des textes, ce qui implique que certains implémentation utilise le NLP. Cette application utilise souvent l'approche statistique (Liddy, 2001). Le TLN est généralement utilisé dans le niveau traitement morphologique.

Elle analyse la variation morphologique des mots ou *stemming*, mais aussi la variation syntaxique (étude des syntagmes nominaux) ainsi que la variation sémantiques : relation liant des lemmes différents mais identiquement proches et la multitude de sens qui peut prendre une forme graphique donnée (polysémie, homonymie) (Tannier, 2006).

Stemming

(Tannier, 2006) La plus courante utilise une approximation des phénomènes linguistiques d'une langue donnée, comme les mécanismes habituels de conjugaison, d'accord ou de genre et en nombre, ou sa dérivation et tente de supprimer les suffixes tout en regroupant les différentes allomorphes (variante graphique d'une même racine).

L'algorithme le plus utilisé est celle de *Lovins* et *Porter* pour la langue anglaise et *Jacques Savoy* pour la langue française.

(Jivani, 2023) analyse de ces algorithmes, et les différentes approche sont illustré dans la Figure 3.7.

L'Extraction d'Information (EI)

Ce domaine d'application est plus récente, qui se base sur la reconnaissance, marquage (tagging), et extraction dans un représentation structurés, certains éléments clés d'information, par exemple : personne, compagnie, localisation, organisation, d'un large collection de texte (Tannier, 2006).

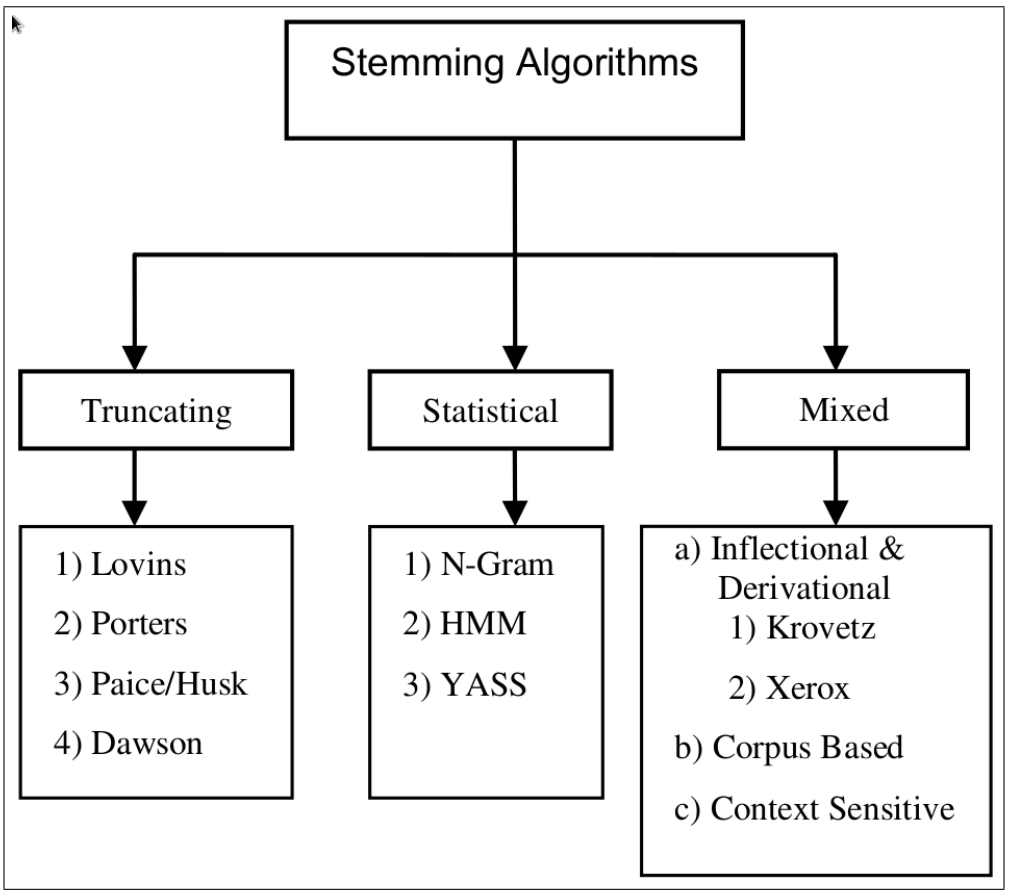

FIGURE 3.7 – Algorithmes de stemming (JIVANI, 2023)

Et d'autres applications

Et d'autres domaines d'applications, comme la machine de traduction (Machine Translation), système de dialogue, système de question-réponse, système de résumé (summarization).

3.12 Conclusion

En conclusion, le Traitement de Langage Naturel est l'un de domaine intéressant, et contribue a beaucoup des recherches dans la domaine de l'informatique. Il est aussi nécessaire dans la recherche d'information. Elle se divise en deux principale branches qui est le traitement de langage écrite et le traitement de langage orale. Elle propose différentes approches et méthodes ainsi que différentes algorithmes pour faire des traitements. Elle est utilisés dans différentes domaines et qui est devenue indissociable de l'intelligence artificielle.

Chapitre 4

Modèle vectoriel

4.1 Introduction

Le modèle vectoriel (Vector Space Model) est l'une de modèle le plus utilisé en recherche d'information, que ce soit textuel (SAADOUNE, 2018) ou multimédia (MARTINET, 2004). Ce modèle est apparu lorsque la pondération binaire est limitant, et que l'appariement partielle n'est pas possible, en introduisant la pondération non binaire et un appariement partielle (BAEZA-YATES et RIBEIRO-NETO, 1999).

Ce modèle a comme principe la modélisation de la requête et des documents sous forme d'un vecteur, pondérer les termes dans ce vecteur avec des méthodes de pondération comme le *TF-IDF*, et calcule ensuite la distance euclidienne entre les vecteurs documents et la vecteur requête pour déterminer la pertinence (YANG, 2000). Un score de similarité est alors calculé pour pouvoir classer les documents jugés pertinent par le système. La représentation du document et la requête est illustré dans la Figure 1.6. Ce modèle est introduit dans la Section 1.9.6.1.

Définition 4.1.1 (Modèle Véctoriel)**.** On note w_{ij} le poids positif et non binaire, du terme i dans le document j qui est associé avec un pair (k_i, d_j) et w_{iq} le poids du terme i dans la requête q. La requête est définie par le vecteur : $\vec{q} = (w_{1q}, w_{2q}, \ldots, w_{tq})$ où t le nombre total des termes d'indexation dans le système. Un document d_j est présenté par le vecteur : $\vec{d_j} = (w_{1j}, w_{2j}, \ldots, w_{tj})$.

La calcul de similarité entre ces deux vecteurs se traduit par la formule :

$$Sim(\vec{d}, \vec{q}) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t}(w_{i,j})^2} \times \sqrt{\sum_{i=1}^{t}(w_{i,q})^2}}$$

Avec $|\vec{d_j}|$ et $|\vec{q}|$ norme du vecteur document et de la vecteur requête.

4.2 Problème de classification

Ce problème est illustré et analyser dans (Baeza-Yates et Ribeiro-Neto, 1999). L'étude de Salton définit la problème de RI comme un problème de classification. Notons une collection de document C et que la requête de l'utilisateur est une vague représentation de l'ensemble des documents A. Le problème est donc de déterminer quelle documents appartient a l'ensemble A, qui est un problème de classification.

Dans un problème de classification, il y a deux problème : la première consiste a déterminer les fonctionnalités qui décrit le mieux les objets dans la collection A ; et la seconde consiste a déterminer les fonctionnalités qui distingue les objets dans collection A pour les autres objets provenant de la collection C. La première fonctionnalité produit la quantification *intra-clustering* tandis que la seconde fonctionnalité produit la quantification *inter-clustering*.

Dans le modèle vectoriel, la similarité *intra-clustering* est quantifié par la mesure de la fréquence du terme k_i dans le document d_j. Cette facteur est souvent appelé facteur *tf* ou *term frequency*, qui décrit la caractérisation *intra-document*. D'autre part, la similarité *inter-clustering* est quantifié par la mesure de l'inverse du fréquence du terme k_i parmi les documents dans la collection. Cette facteur est souvent appelé facteur *idf* ou *inverse document frequency*

Pour avoir un meilleur classification des documents, il faut balancer ces deux mesures.

4.3 Méthodes de pondération des termes

Cette facteur est permet de calculer la fréquence d'un terme dans un document, voir Section 1.10.

Définition 4.3.1. Cette définition est tiré de *Modern Information Retrieval* (Baeza-Yates et Ribeiro-Neto, 1999). Notons N le nombre total des documents dans le système et n_i le nombre de documents contenant le terme k_i. Notons $freq_{i,j}$ la fréquence du terme k_i dans le document d_j (le nombre de dois où le terme k_i est mentionné dans le texte du document d_j). Alors la fréquence normalisé $f_{i,j}$ du terme k_i dans le document d_j est donné par

$$f_{i,j} = \frac{freq_{i,j}}{\max_l freq_{i,j}}$$

où le maximum est calculé a travers les termes qui sont mentionné dans le texte du document d_j. Si le terme k_i n'appartient pas au document d_j alors $f_{i,j} = 0$. D'autre part, notons idf_i l'*inverse document frequency* pour le terme k_i, qui est donné par

$$idf_i = \log \frac{N}{n_i}$$

La pondération de terme le plus connu et plus utilisé est donné par

$$w_{i,j} = f_{i,j} \times \log \frac{N}{n_i}$$

ou par la variation de ce formule. Ce pondération de termes est aussi appelé facteur *tf-idf*

Certains variations de ce méthode de pondération est introduit en 1988, mais la formule précédente est efficace pour la plupart des collection. Pour la requête, Salton et Buckley (Baeza-Yates et Ribeiro-Neto, 1999) a suggéré la pondération suivante

$$w_{i,q} = \left(0.5 + \frac{0.5 freq_{i,q}}{\max_l freq_{i,j}}\right) \times \log \frac{N}{n_i}$$

4.4 Variant du TF-IDF

Les variants de ces méthodes sont illustré dans la Tableau 4.1 qui est celle de la *tf*, la Tableau 4.2 qui est celle de l'*idf* et la Tableau 4.3 qui est celle de *tf-idf*.

Weighting sheme	**TF weight**
Binary	0, 1
Raw count	$f_{t,d}$
Term frequency	$f_{t,d} / \sum_{t' \in d} f_{t',d}$
Log normalization	$\log(1 + f_{t,d})$
Double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{t' \in d}} f_{t',d}$
Double normalization K	$K + K \cdot \frac{f_{t,d}}{\max_{t' \in d}} f_{t',d}$

Table 4.1 – Variant du TF (SAADOUNE, 2018)

Weighting sheme	**IDF** $(n_i = \lvert\{d \in D : t \in d\}\rvert)$
Unary	1
Inverse document frequency	$\log \frac{N}{n_t} = -log \frac{n_t}{N}$
Inverse document frequency smooth	$\log\left(\frac{N}{1+n_t}\right) + 1$
Inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1+n_t}\right)$
Log normalization-idf	$(1 + \log f_{t,d}) \cdot \log \frac{N}{n_t}$

Table 4.2 – Variant de l'IDF (SAADOUNE, 2018)

Weighting sheme	**TF-IDF**
Count-idf	$f_{t,d} \cdot \log \frac{N}{n_t}$
Double normalization-idf	$\left(0.5 + 0.5 \cdot \frac{f_{t,q}}{\max_t f_{t,q}}\right) \log \frac{N}{n_t}$
Log normalization-idf	$(1 + \log f_{t,d}) \cdot \log \frac{N}{n_t}$

TABLE 4.3 – Variant du TF-IDF (SAADOUNE, 2018)

4.5 Document inversé

Un *document inversé* ou *fichier inversé* est un structure de données au cœur des moteurs de recherche gigantesque, des réseaux sociaux et des architectures de stockage. Il permet de stocker des millions des documents (PIBIRI et VENTURINI, 2020).

Définition 4.5.1 (Fichier inversé)**.** Considérons une collection des documents textuels, et que chacun est décrit comme un ensemble des termes. Pour chaque terme distinct t qui apparaissant dans la collection, un séquence d'entier S_t est crée et répertorie, dans l'ordre trié, tous les identifiants des documents (dorénavant, docIDs) où le terme apparaît. La séquence S_t est appelé liste inversé ou *posting list* pour le terme t (PIBIRI et VENTURINI, 2020).

Le fichier inversé peut stocker des informations additionnel pour chaque terme, comme les positions du termes dans la document, le nombre d'occurrence du terme dans le document (fréquences).

Un exemple d'un fichier inversé est illustré dans la Figure 4.1.

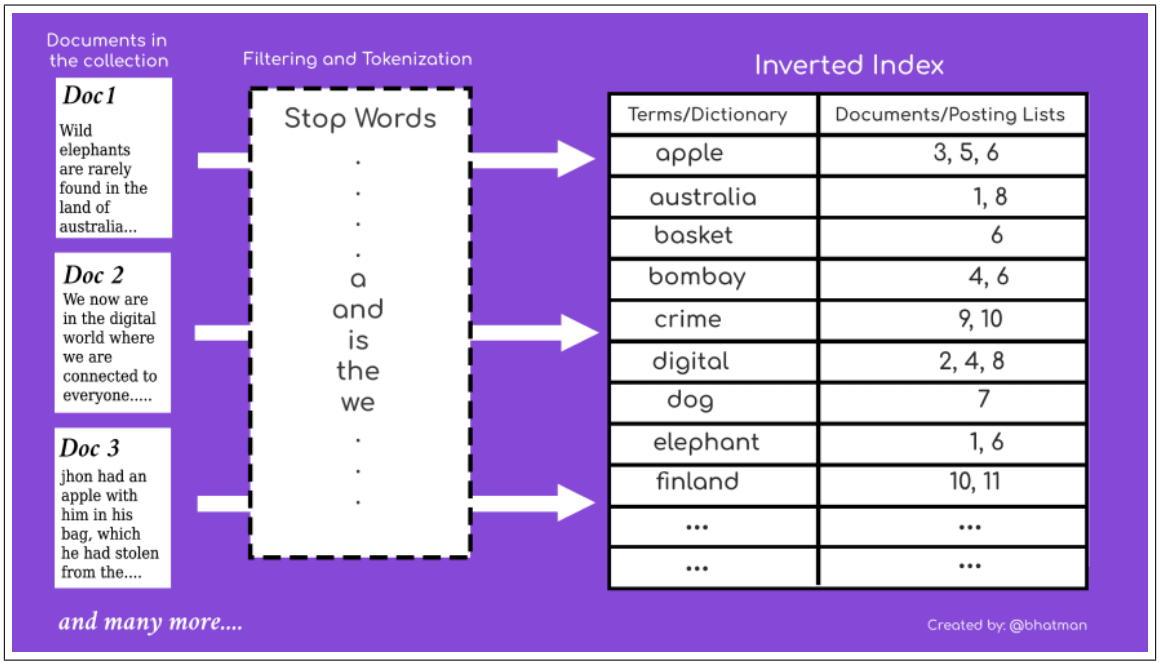

FIGURE 4.1 – Variant du TF-IDF (BHAT, IMRAN, 2023)

4.6 Avantages

Le modèle vectoriel ont les avantages suivantes (SOULIER, 2014 ; ZIANI, 2022) :

— **Facile a mettre en œuvre** : ce modèle est facile a implémenter en informatique, grâce a ces différentes formules ainsi que le fichier inversé.
— **Appariement partielle** : ce modèle propose la possibilité de faire une appariement approximatif avec un degré de pertinence. En d'autre terme, l'utilisateur reçoit des documents qui pourra satisfaire la moitié de la requête ou en quelque pourcentage.
— **Organisation des résultats** : comme l'appariement partielle est possible, les documents retournés sont alors organisé selon leur degré de pertinence décroissant afin de faciliter la selection des documents par l'utilisateur. L'utilisateur passe moins de temps donc a juger les résultats.
— **Définir une limite de pertinence** : ainsi ce modèle permet de définir une seuil pour calculer la pertinence afin que les documents ayant la pertinence inférieur a ce seuil ne sont pas retournés.

C'est grâce a ces avantages que ce modèle est populaire, et qu'il est le plus utilisé par les moteurs de recherche actuel.

4.7 Inconvénients

Par contre ce modèle a certaines lacunes (BAEZA-YATES et RIBEIRO-NETO, 1999 ; Vaibhav SINGH et Vinay SINGH, 2022) :

— **Indépendance des termes d'indexation** : ce qui implique que la notion sémantique du document est perdu. Mais ce problème a été solutionné par la mise place de regroupement des termes qui ont la même sens, on l'appelle *N-grammes*. Ou bien une autre approche est d'utiliser le modèle d'indexation sémantique latente (Latent Semantic Index).
— **Pas de théorie réelle** : il n'existe pas de base théorique réelle pour l'hypothèse d'un espace de termes.
— **Pondération des termes** : le poids associé aux vecteurs est totalement arbitraire, et que c'est un système indépendant.

4.8 Conclusion

Le modèle vectoriel est l'un de modèle le plus populaire dans la recherche d'information que ce soit textuel ou multimédia. Ce modèle est populaire grâce a son système de pondération non binaire (*tf, idf, tf-idf*), la possibilité de faire une appariement partielle ainsi que le classement des résultats. Ce modèle propose différentes mesure de similarité pour l'appariement *document-requête*, ainsi des variation pour le système de pondération. Le modèle vectoriel stock les documents dans le fichier inversé pour appliquer la recherche.

Chapitre 5

Simulation et Réalisation

5.1 Conception du logiciel

5.1.1 Description détaillée de la conception du logiciel

5.1.1.1 Architecture logicielle

Ce système (SRI) est un système orienté WEB utilisant l'architecture MVC ou Modèle Vue Controlleur. C'est un motif d'architecture logicielle destiné aux interfaces graphiques, lancé en 1978 et très populaire pour les applications web (Wikipedia contributors, s. d.).

Cet architecture est illustré dans la Figure 5.1

Le Modèle (Model)

Le Modèle contient les données a afficher a l'utilisateur. En d'autre terme, c'est la representation de la base de données. Dans le cadre de ce système, l'index inversé, les documents ainsi que les catégorie des documents sont representé par un modèle chacun.

La Vue (View)

Elle contient la representation graphique, tel que les formulaires, les listes, les tables, etc. Elle est généralement ecrite en langage HTML pour une application WEB. Elle represente la formulaire de recherche, ainsi que les différenes filtres de recherche par l'utilisateur, et aussi les parties concernant l'authentification et d'autres.

Le Controlleur (Controller)

Il contient la logique concernant les actions effectuées par l'utilisateur. En d'autre terme, il est le chef d'orchestre du système, traite les informations provenant de l'utilisateur et les persister dans la base de données et de recuperer des données et de les afficher a l'utilisateur. Dans

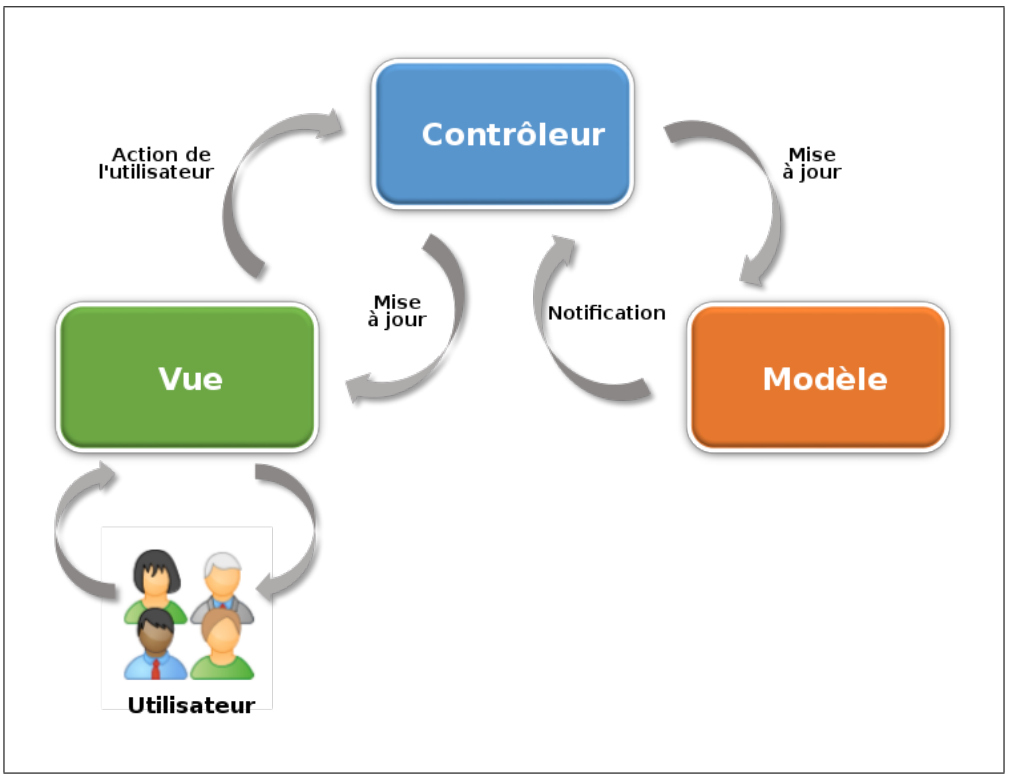

FIGURE 5.1 – Architecture MVC (WIKIPEDIA CONTRIBUTORS, s. d.)

ce système, il est responsable de tous traitement de documents, authentification, et d'autres traitements comme le telechargement et protection des documents.

Il est utilisé par de nombreux frameworks pour applications web tels que Ruby on Rails, Grails, ASP.NET MVC, Spring, Struts, Symfony, Apache Tapestry, Laravel, AdonisJS, Django ou AngularJS.

5.1.1.2 Choix de conception clés

Pour ce système, on distingue deux type de conception tel que la *conception de la base de données* et la *conception de l'interface utilisateur*. Pour la base de données, deux approche sont possible tel que *MERISE* et *UML*. L'une de grande différence entre MERISE et UML ce que MERISE traite les données et les traitements séparement tandis que UML non.

MERISE

Merise est une méthode informatique dédiée à la modélisation qui analyse la structure à informatiser en terme de systèmes. Le gros avantage de cette méthode est qu'elle permet de cadrer le projet informatique et de *discuter* en se comprenant entre utilisateurs et informaticiens (BASE-DE-DONNEES.COM, s. d.).

Créée dans les années 70 sur commande de l'État français et destinée aux gros projets informatiques de l'époque, la méthode a perduré jusqu'à aujourd'hui. Son utilisation très répandue en Europe constitue un socle difficilement contournable lorsque l'on s'attache à la création de

bases de données.

Merise est en fait un outil analytique qui facilite la création de base de données et de projets informatique. Le principal auteur de la méthode est Hubert Tardieu qui se basa sur les travaux autour du modèle relationnel de Codd.

Elle permet réelement de :

— décrire le fonctionnement du système à informatiser tel que les **données** representé par le MCD (Modèle Conceptuel de Données) qui determine les relations et les dépendances entres les différents acteurs (utilisateur, administrateur, documents), et les *traitements* representé par le MCT (Modèle Conceptuel de Traitement) qui determine comment les acteurs travaillent-ils ensemble.
— proposer une implémentation logique tel que MLD (Modèle Logique de Données), MLT (Modèle Logique de Traitement).
— Proposer une construction concrète et utilisable du point précédent tel que la MPD (Modèle Physique de Donnée).

UML

Le Langage de Modélisation Unifié, de l'anglais Unified Modeling Language, est un langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet. L'UML est une synthèse de langages de modélisation objet antérieurs : Booch, OMT, OOSE (UML Organization, s. d.).

Principalement issu des travaux de Grady Booch, James Rumbaugh et Ivar Jacobson, UML est à présent un standard adopté par l'Object Management Group. UML 1.0 a été normalisé en janvier 1997 ; UML 2.0 a été adopté par l'OMG en juillet 2005.

UML propose différentes diagrammes, et principalement catégorisé en trois catégories : Bien sûr, voici une explication détaillée de chaque catégorie de diagrammes UML :

1. **Structure** : Où on trouve
   — *Diagramme de classes* : Il montre la structure statique d'un système en mettant l'accent sur les classes du système, leurs attributs, leurs opérations et les relations entre les classes.
   — *Diagramme d'objets* : Il illustre des exemples spécifiques d'objets et de relations entre ces objets, montrant une instance particulière d'un diagramme de classes à un moment donné.
   — *Diagramme de composants* : Il met l'accent sur les composants d'un système logiciel et leurs dépendances, en montrant la structure des composants et la manière dont ils interagissent au niveau de l'architecture.

— *Diagramme de paquetages* : Il organise les éléments d'un modèle en groupes, montrant comment ces éléments sont regroupés en paquets et les relations entre les paquets.

2. Comportément
   — Diagramme de cas d'utilisation : Il décrit les interactions entre les utilisateurs et un système donné, mettant l'accent sur les fonctionnalités offertes par le système du point de vue de l'utilisateur.
   — Diagramme de séquence : Il montre l'interaction entre les objets, en mettant l'accent sur la séquence temporelle des messages échangés entre les objets lors de l'exécution d'un scénario particulier.
   — Diagramme d'activités : Il représente le flux de contrôle d'une activité ou d'un processus, mettant l'accent sur les actions et les décisions qui composent le processus.
   — Diagramme d'états-transitions : Il modélise les transitions d'états pour un objet ou une entité donnée, montrant comment l'objet réagit aux événements au fil du temps.
3. Déploiement
   — Diagramme de déploiement : Il montre la configuration matérielle d'un système et la manière dont les composants logiciels sont déployés sur cette configuration matérielle.

Ces différentes catégories de diagrammes UML offrent des moyens visuels efficaces pour modéliser les différents aspects d'un système logiciel, ce qui facilite la compréhension et la communication entre les différentes parties prenantes impliquées dans le processus de développement logiciel.

Dans le cadre de ce dévoir, on utilisera l'approche UML, vu qu'on utilise une architecture MVC et une approche orienté objet (OOP), cette approche sera plus bénéfique. Mais il est possible et ce sera envisageable d'utiliser Merise pour ce système.

Pour la conception de l'interface utilisateur, le langage de balisage HTML5, le feulle de style CSS3 ainsi que le langage JavaScript est utilisé.

SGBD

Un SGBD ou Système de Gestion de Base de Données est un logiciel système permettant aux utilisateurs et programmeurs de créer et de gérer des bases de données. Plus précisement, il permet a un ordinateur de stocker, récupérer, ajouter, supprimer et modifier des données. Ce système est aujourd'hui utilisé dans presque tous les outils que nous utilisons (Oracle, s. d.).

Le SGBD gère trois choses importantes : les *données*, le *moteur de base de données* qui permet d'accéder aux données, de les verrouiller et de les modifier, et le *schéma de base de données*, qui définit la structure logique de la base de données. Ces trois éléments fondamentaux contribuent à assurer la concomitance, la sécurité, l'intégrité des données et l'uniformité des procédures administratives (Oracle, s. d.).

5.1.1.3 Les fonctionnalités de base

Citons les fonctionnalités de base qui sont le plus importants pour ce système.

1. **Indexer des documents** : capacité d'indexer des documents textuelles de différente format tel que PDF, WORD et peut être des images donténant des textes. Cette fonctionnalité inclus l'extraction automatique de titre, resumé, date de soutenance, ainsi de faire un traitement de langage naturel. Cette donctionnalité se fait par un upload de fichier dans un navigateur WEB.
2. **Système d'authentification** : permet a l'utilisateur de s'authentifier ou se créer des comptes afin de deposer et telecharger des documents. Elle permet aussi aux amdinistrateurs de valider des documents, les supprimer si necessaire. Elle permet aussi de disposer d'un espace membre en tant qu'étudiant et en tant qu'enseignant, verouller des documents et de ne donner l'accès qu'a certains utilisateurs.
3. **Recherche par mots clés** : ette fonctionnalité permet aux utilisateurs de faire des recherches en utilisant une zone de saisie par des mots clés afin de satisfaire ses besoins. Cette fonctionnalité inclus d'autres fonctionnalités tel que le *système de filtre de recherche* : recherche par conténu, par titre ou par auteur ; *système de filtre d'organisation* : par année croissante ou décroissante, par université ; *traitement de la reqête* : suppréssion des mots vides, correction des orthographes, pondération du terme de la requête. Elle se charge de faire l'appariément entre la requête de l'utilisateur et les documents dans la collection, et afficher ensuite les résultats par ordre de pertinence décroissante. L'utilisateur peut avoir une suggestion des mots clés lorsqu'il exprime son besoin d'information.
4. **Page de resultat et visualisation de document** : permet d'afficher les resultats avec des paginations, ainsi que la possibilité de visualiser le resumé ou la partie du document. Il est aussi possible de visualiser un document en entier.

5.1.2 Technologies et outils utilisés

5.1.2.1 Technologies

Pour devellopер ce SRI, on a utilisé des téchnologies orientés vers l'intelligence artificiel implementé dans un application web. On aura utilisé une base de données pour stocker les données tel que les documents, les informations de l'utilisateur, etc. Ci-dessous la liste des téchnologies utilisés pour la réalisation de ce système.

— **Framework Django** : est un framework web developpé en langage Python tilisant l'architecture MVT (Model-View-Template). A bien noter que dans Django, View est l'equivalent d'un Controlleur et que Template est l'equivalent de Vue. Ce choix est dû au fait qu'il est facile d'integré des librairies d'intelligence artificiel dans ce framework, ainsi que les fonctionnalités pré-concus facilitera beaucoup le devellopement (Django Software Foundation, s. d.).
— **MySQL** : pour stocker les données comme on a dit ci-dessus. Elle est gratuite et largement

suffisant pour ce genre de système en terme de capacité de stockage et en terme de rapidité de récuperation des données (MySQL, s. d.).

— **Spacy** : est un librairy python permettant de faire un traitement de langage naturel. Il utilise un modèle de machine learning pré-entrainé pour différentes langages. Comme pour la langue française par exemple il y a les modèle *frcorewebnewssm*. Ce librairie permet de segmenter des textes en français, supprimer les mots vides, et recupérer la racine des mots, c'est ce librairie qui se charge de l'indexation dans ce SRI (Explosion AI, s. d.).
— **NLTK** : ou Natural Language Toolkit est aussi ule librairie de traitement de langage naturel (NLTK Project, s. d.)
— Numpy (NumPy Community, s. d.)
— ScikitLearn (scikit-learn contributors, s. d.)
— PDFPlumber (JSvine, s. d.)
— FItz library (*PyMuPDF (fitz)* s. d.)
— Spellchecker library (*Spellchecker* s. d.)

5.1.2.2 Langage de programmation

Description des langages de programmation utilisés pour la mise en œuvre du logiciel, par exemple, Python, Java, ou JavaScript. Justification de l'utilisation de ces langages en fonction de leur adaptabilité aux exigences spécifiques du logiciel.

— Python (Python Software Foundation, 2023)
— SQL

5.1.2.3 Outils

Présentation des outils de développement utilisés, comme les environnements de développement intégrés (IDE), les systèmes de contrôle de version, et les outils de test. Explication de la façon dont ces outils ont facilité le processus de conception et de développement du logiciel.

— Visual Studio Code (Explication de ses diverses extensions) (Microsoft, s. d.)
— Git et Github (Système de controle de version / Explication de la facilité de suivre des modifications) (Git Community, s. d. ; GitHub, s. d.)
— Machine utilisé pour le développement (CPU, GPU, RAM, HDD, OS)
— Outil de test de django

5.2 Développement du logiciel

5.2.1 Présentation des différentes étapes du processus de développement

5.2.1.1 Planification

Explication du processus de planification, y compris l'identification des fonctionnalités clés, la répartition des tâches et la définition des jalons du projet. Discussion sur la manière dont la planification a facilité le développement du logiciel.

5.2.1.2 Programmation

Description des étapes de programmation, y compris la mise en œuvre des fonctionnalités principales du logiciel et la résolution des problèmes techniques rencontrés lors du processus de développement. Illustration de certains segments de code pertinents pour la compréhension du développement.

5.2.1.3 Tests

Présentation de la stratégie de test adoptée pour évaluer le bon fonctionnement du logiciel. Description des tests unitaires, des tests d'intégration et des tests de validation effectués pour assurer la qualité du logiciel.

5.2.2 Défis rencontrés lors du développement et solutions mises en œuvre

5.2.2.1 Contraintes techniques

Discussion sur les problèmes techniques spécifiques rencontrés lors du développement et des solutions techniques mises en place pour les surmonter. Analyse des obstacles et des stratégies d'atténuation adoptées pour assurer la progression continue du projet.

5.2.2.2 Gestion des délais

Évaluation des défis liés à la gestion du temps et des échéances du projet. Présentation des techniques de gestion de projet mises en place pour respecter les délais et assurer la livraison du logiciel dans les délais impartis.

5.2.2.3 Collaboration d'équipe

Discussion sur les enjeux liés à la collaboration au sein de l'équipe de développement et des mesures prises pour favoriser une communication efficace et une coordination harmonieuse entre les membres de l'équipe.

5.3 Tests et validation du logiciel

5.3.1 Stratégies de test utilisées pour évaluer la performance et la fonctionnalité du logiciel

5.3.1.1 Tests unitaires

Description des tests unitaires réalisés pour vérifier le bon fonctionnement des composants individuels du logiciel.

5.3.1.2 Tests d'intégration

Présentation des tests d'intégration effectués pour évaluer l'interopérabilité des différents modules du logiciel.

5.3.1.3 Tests de validation

Explication des tests de validation effectués pour vérifier si le logiciel répond aux exigences spécifiées dans le cahier des charges.

5.3.2 Résultats des tests et analyse critique des performances du logiciel

5.3.2.1 Résultats des tests

Présentation des résultats détaillés des tests effectués, y compris les rapports de tests, les captures d'écran et les données de performance.

5.3.2.2 Identification des problèmes

Discussion sur les éventuels problèmes ou erreurs identifiés au cours des tests, avec analyse critique des mesures prises pour corriger ces problèmes.

5.4 Implémentation du logiciel dans un contexte réel

5.4.1 Processus d'implémentation

Explication du processus d'implémentation du logiciel dans un environnement réel, en mettant l'accent sur les étapes clés et les considérations spécifiques prises en compte lors du déploiement.

5.4.2 Évaluation de l'efficacité de l'implémentation

Analyse de l'efficacité de l'implémentation du logiciel en fonction des objectifs initiaux et des résultats observés dans le contexte réel.

5.4.3 Rétroaction des utilisateurs

Présentation des retours d'expérience et des commentaires des utilisateurs finaux concernant l'utilisation et les performances du logiciel dans leur environnement réel.

5.4.4 Réponses aux problèmes éventuels

Discussion sur la manière dont les problèmes éventuels rencontrés lors de l'implémentation ont été résolus, et sur les ajustements apportés pour améliorer l'expérience utilisateur et les performances du logiciel.

5.5 Évaluation des performances et des résultats

5.5.1 Évaluation critique des performances du logiciel

Analyse approfondie des performances du logiciel en termes de fonctionnalité, de convivialité, de fiabilité et d'efficacité, en se référant aux critères établis au début du projet.

5.5.2 Comparaison avec les objectifs initiaux

Comparaison des performances réelles du logiciel avec les objectifs initiaux définis lors de la conception, en mettant en évidence les écarts éventuels et les facteurs contributifs.

5.5.3 Identification des forces et des faiblesses

Identification des points forts et des limitations du logiciel en fonction des retours d'expérience des utilisateurs et des évaluations techniques, en mettant l'accent sur les domaines nécessitant d'éventuelles améliorations ou ajustements.

5.5.4 Implications et recommandations

Discussion des implications des résultats obtenus, avec des recommandations pratiques pour l'amélioration continue du logiciel ainsi que des suggestions pour des développements futurs ou des extensions potentielles.

5.6 Réflexions sur l'expérience de développement

5.6.1 Analyse critique des leçons apprises

Analyse réfléchie des principaux enseignements tirés de l'expérience de développement du logiciel, en mettant l'accent sur les succès, les défis et les stratégies d'adaptation ou d'amélioration.

5.6.2 Évaluation de l'efficacité des stratégies de développement

Évaluation de l'efficacité des différentes stratégies de développement utilisées tout au long du processus, en mettant en évidence les approches qui ont bien fonctionné et celles qui auraient pu être améliorées.

5.6.3 Suggestions pour des améliorations futures

Proposition de suggestions concrètes pour améliorer les processus de développement, les stratégies de gestion de projet et les approches techniques pour de futurs projets similaires.

5.6.4 Perspectives de recherche future

Discussion sur les pistes de recherche future potentielles basées sur les lacunes identifiées dans le cadre du projet actuel, en mettant l'accent sur les domaines qui méritent une exploration plus approfondie et des développements ultérieurs.

Conclusion générale

Nulla in ipsum. Praesent eros nulla, congue vitae, euismod ut, commodo a, wisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nonummy magna non leo. Sed felis erat, ullamcorper in, dictum non, ultricies ut, lectus. Proin vel arcu a odio lobortis euismod. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Proin ut est. Aliquam odio. Pellentesque massa turpis, cursus eu, euismod nec, tempor congue, nulla. Duis viverra gravida mauris. Cras tincidunt. Curabitur eros ligula, varius ut, pulvinar in, cursus faucibus, augue.

Nulla mattis luctus nulla. Duis commodo velit at leo. Aliquam vulputate magna et leo. Nam vestibulum ullamcorper leo. Vestibulum condimentum rutrum mauris. Donec id mauris. Morbi molestie justo et pede. Vivamus eget turpis sed nisl cursus tempor. Curabitur mollis sapien condimentum nunc. In wisi nisl, malesuada at, dignissim sit amet, lobortis in, odio. Aenean consequat arcu a ante. Pellentesque porta elit sit amet orci. Etiam at turpis nec elit ultricies imperdiet. Nulla facilisi. In hac habitasse platea dictumst. Suspendisse viverra aliquam risus. Nullam pede justo, molestie nonummy, scelerisque eu, facilisis vel, arcu.

Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam congue neque id dolor.

Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh. Maecenas non sem quis tortor eleifend fermentum. Etiam id tortor ac mauris porta vulputate. Integer porta neque vitae massa. Maecenas tempus libero a libero posuere dictum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean quis mauris sed elit commodo placerat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl. Vestibulum sed nisl eu sapien cursus rutrum.

Nulla non mauris vitae wisi posuere convallis. Sed eu nulla nec eros scelerisque pharetra.

Nullam varius. Etiam dignissim elementum metus. Vestibulum faucibus, metus sit amet mattis rhoncus, sapien dui laoreet odio, nec ultricies nibh augue a enim. Fusce in ligula. Quisque at magna et nulla commodo consequat. Proin accumsan imperdiet sem. Nunc porta. Donec feugiat mi at justo. Phasellus facilisis ipsum quis ante. In ac elit eget ipsum pharetra faucibus. Maecenas viverra nulla in massa.

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.

Bibliographie

A. Rozenknop (s. d.). *Modèles en Recherche d'Information*. Cours Master Recherche Paris 13, Recherche et extraction d'information.

Abu-Salih, Bilal (2018). « Applying Vector Space Model (VSM) Techniques in Information Retrieval for Arabic Language. » In : *CoRR* abs/1801.03627. URL : http://dblp.uni-trier.de/db/journals/corr/corr1801.html#abs-1801-03627.

Andrea Ferrario, Mara Nägelin (jan. 2020). *The Art of Natural Language Processing: Classical, Modern and Contemporary Approaches to Text Document Classification*.

B S Harish D S Guru, S Manjunath (juill. 2010). *Representation and Classification of Text Documents: A Brief Review*.

Baeza-Yates, Ricardo A. et Berthier A. **Ribeiro-Neto** (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley. ISBN : 0-201-39829-X. URL : http://www.ischool.berkeley.edu/~hearst/irbook/glossary.html.

Base-de-donnees.com (s. d.). *Base-de-donnees.com*. https://www.base-de-donnees.com/merise/. Accessed: November 7, 2023.

Bellaouar, Slimane, Mahieddine **Djoudi** et Samir **Zidat** (2009). « Construction et utilisation d'un thésaurus pour la recherche d'information sur le web ». In : *Conférence Internationale des Technologies de l'Information et de la Communication, CITIC'09, Sétif, Algérie*.

Benayache, Ahcène (2005). « Construction d'une mémoire organisationnelle de formation et évaluation dans un contexte e-learning ». Thèse de doctorat. Thèse de doct. Université de Technologie de Compiègne.

Bhat, Imran (2023). *Introduction to Inverted Indexes*. Accessed on 2023-09-08. URL : https://dev.to/im_bhatman/introduction-to-inverted-indexes-l04.

Blair, David C. et M. E. **Maron** (mars 1985). « An Evaluation of Retrieval Effectiveness for a Full-Text Document-Retrieval System ». In : *Commun. ACM* 28.3, p. 289-299. ISSN : 0001-0782. DOI : 10.1145/3166.3197. URL : https://doi.org/10.1145/3166.3197.

Boubée, Nicole et André **Tricot** (2010). *Qu'est-ce que rechercher de l'information? Nouvelle édition*. DOI: https://doi.org/10.4000/books.pressesenssib.799. Villeurbanne : Presses de

l'enssib. ISBN : 9782375460412. URL : http://books.openedition.org/pressesenssib/799.

Boughanem, Mohand (s. d.). *Evaluation des performances dans les SRI*. URL : https://www.irit.fr/~Mohand.Boughanem/slides/RI/chap11-Evaluation.pdf.

Dale, Robert et Moisl, Hermann et Somers, Harold (s. d.). *Handbook for Natural Language Processing*.

Devopedia (2023). *Natural Language Processing*. Accessed on 2023-09-08. URL : https://devopedia.org/natural-language-processing.

Django Software Foundation (s. d.). *Django*. https://www.djangoproject.com/. Accessed: November 7, 2023.

Dr. LOUNNAS Bilal (2023). *Information retrieval (IR)*. Coours de recherche d'information, Université Mohamed Boudiaf - M'Sila.

Explosion AI (s. d.). *spaCy*. https://spacy.io/. Accessed: November 7, 2023.

Git Community (s. d.). *Git - version control system*. https://git-scm.com/. Accessed: November 7, 2023.

GitHub (s. d.). *GitHub*. https://github.com/. Accessed: November 7, 2023.

Hiemstra, Djoerd (1999). *Model de recherche d'information*.

Jérôme VELO (2009). « Interactions entre un utilisateur et un systeme informatique : application a un systeme d'e-administration ». Thèse de doctorat en cotutelle pour l'obtention du grade de docteur en informatique des universites de rouen et d'antananarivo. Thèse de doct. Université d'Antananarivo.

Jivani, Anjali Ganesh (2023). « A Comparative Study of Stemming Algorithms ». In : *Journal of Natural Language Processing* X.Y, Z.

Journal du Net (2023). *Moteur de recherche : définition, traduction et acteurs*. URL : https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203539-moteur-de-recherche-definition-traduction-et-acteurs/.

JSvine (s. d.). *pdfplumber*. https://github.com/jsvine/pdfplumber. Accessed: November 7, 2023.

Kien QUACHTAT (2012). « Recherche d'information sur le WEB et moteurs de recherche: le cas des lycéens ». Thèse de doctorat. Ecole normal supérieur de CACHAN.

LeBigData (2023). *Base de données*. URL : https://www.lebigdata.fr/base-de-donnees.

Liddy, Elizabeth D (2001). « Natural language processing ». In.

Martinet, Jean (déc. 2004). « Un modèle vectoriel relationnel de recherche d'information adapté aux images ». Theses. Université Joseph-Fourier - Grenoble I.

Mazari, Ahmed Cherif (2022). « Contribution à l'amélioration de la recherche d'information par utilisation des méthodes sémantiques: application à la langue arabe ». Thèse de doct. Université de Mohamed Kheider Biskra.

Microsoft (s. d.). *Visual Studio Code*. `https://code.visualstudio.com/`. Accessed: November 7, 2023.

Mozilla Developer Network (2023). *Search Engine*. URL : `https://developer.mozilla.org/fr/docs/Glossary/Search_engine`.

MySQL (s. d.). *MySQL*. `https://www.mysql.com/fr/`. Accessed: November 7, 2023.

NLTK Project (s. d.). *NLTK*. `https://www.nltk.org/`. Accessed: November 7, 2023.

NumPy Community (s. d.). *NumPy*. `https://numpy.org/`. Accessed: November 7, 2023.

Oracle (2023). *What is Database*. URL : `https://www.oracle.com/fr/database/what-is-database/`.

— (s. d.). *Oracle Database: Système de Gestion de Base de Données (SGBD) - Définition*. `https://www.oracle.com/fr/database/systeme-gestion-base-de-donnees-sgbd-definition/`. Accessed: November 7, 2023.

Paradis, Francois (nov. 1996). « Un modèle d'indexation pour les documents textuels structurés ». Theses. Université Joseph-Fourier - Grenoble I. URL : `https://theses.hal.science/tel-00005009`.

Pibiri, Giulio Ermanno et Rossano **Venturini** (2020). « Techniques for inverted index compression ». In : *ACM Computing Surveys (CSUR)* 53.6, p. 1-36.

PyMuPDF (fitz) (s. d.). `https://pymupdf.readthedocs.io/en/latest/index.html`. Accessed: November 7, 2023.

Python Software Foundation (2023). *Python Documentation*. Accessed: November 7, 2023. URL : `https://www.python.org/`.

R Manikandan, Dr. R Sivakumar (mars 2018). *Machine learning algorithms for text-documents classification: A review*.

Ranks NL (2023). *French Stopwords*. Accessed: 03/08/2023. URL : `https://ranks.nl/stopwords/french`.

SAADOUNE, Billal (2018). « Development of search engine based on vector space model ». Mém. de mast. UNIVERSITY OF MOHAMED BOUDIAF - M'SILA. URL : `http://dspace.univ-msila.dz:8080//xmlui/handle/123456789/15946`.

Salton, Gérard (1989). « Traitement automatique de texte : la transformation, l'analyse et la récupération de l'information par un ordinateur ». In : *Lecture : Addison-Wesley* 169.

Salton, Gerard et Fox, Edward A et Wu, Harry (1983). « Récupération d'informations booléennes étendues ». In : *Communications de l'ACM* 26.11. Sous la dir. de États-Unis **ACM New York** NY, p. 1022-1036.

Salton, Gerard et Wong, Anita et Yang, Chung-Shu (1975). « Un modèle d'espace vectoriel pour l'indexation automatique ». In : *Communications de l'ACM* 18.11. Sous la dir. de États-Unis **ACM New York** NY, p. 613-620.

scikit-learn contributors (s. d.). *scikit-learn*. `https://scikit-learn.org/`. Accessed: November 7, 2023.

Singh, Vaibhav et Vinay **Singh** (juill. 2022). *VECTOR SPACE MODEL: AN INFORMATION RETRIEVAL SYSTEM*.

Soulier, Laure (2014). « Définition et évaluation de modèles de recherche d'information collaborative basés sur les compétences de domaine et les rôles des utilisateurs ». Français. ffNNT: ff. fftel-01110721f. Thèse de doct. Université de Toulouse. URL : `https://tel.archives-ouvertes.fr/tel-01110721/document`.

Spellchecker (s. d.). `https://pypi.org/project/pyspellchecker/`. Accessed: November 7, 2023.

Tannier, Xavier (2006). « Traitement automatique du langage naturel pour l'extraction et la recherche d'informations ». In : *Ecole Nationale Supérieure des Mines de Saint-Etienne* 16, p. 17.

Theses.fr (2023). *Theses.fr*. Accessed: 03/08/2023. URL : `https://www.theses.fr/112289355`.

UML Organization (s. d.). *Unified Modeling Language*. `http://uml.org/`. Accessed: November 7, 2023.

Université d'Antananarivo (2016). *Thèse malgache en ligne*. Thèses soutenu dans les 6 Universités de Madagascar dépuis 2002.

Wikipedia contributors (s. d.). *Modèle-vue-contrôleur*. `https://fr.wikipedia.org/wiki/ModÃÍle-vue-contrÃťleur`. Accessed: November 7, 2023.

Yang, Kiduk (2000). « Information Retrieval on the Web ». In : *ACM Computing Surveys*, n. pag.

ZIANI, Chaima (2022). « Réalisation d'un moteur de recherche avec une approche sémantique ». Mém. de mast. UNIVERSITY OF MOHAMED BOUDIAF - M'SILA. URL : `https://dspace.univ-bba.dz:443/xmlui/handle/123456789/3652`.