الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Systèmes Informatiques

Thème :

FILTRAGE DES TAGS DANS UN ENVIRONNEMENT COLLABORATIF

Encadré Par :

Dr. Boughareb Djalila

Présenté par :

Zeghoum Chouaib

Juillet 2019

Remerciements

En tout premier lieu, je remercie le bon Allah Tout Puissant, qui m'a donné la force, la volonté et le courage pour terminer ce modeste travail

Je tiens à exprimer toute ma reconnaissance à ma directrice de mémoire, **Dr. Boughareb Djalila.** *Je la remercie de m'avoir encadré, orienté, aidé et conseillé.*

J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions durant mes recherches.

Je remercie mes très chers parents, qui ont toujours été là pour moi. Je remercie ma sœur, pour leurs encouragements.

Enfin, je remercie mes amis Tamer Mahboubi et Imad chebata qui ont toujours été là pour moi. Leur soutien inconditionnel et leurs encouragements ont été d'une grande aide.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

Dédicace

Je dédie ce travail :

A mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études.

A mes chères sœurs, pour leurs encouragements permanents, et leur soutien moral.

A toute ma famille pour leur soutien tout au long de mon parcours universitaire.

A mes chers amis, pour leur appui et leur encouragement.

A tous mes collègues de promotion.

Enfin, a tousceux qui m'estiment.

Résumé

La grande quantité de texte disponible sous forme numérique peut rendre difficile l'accès efficace aux informations. Classer les textes est un bon moyen d'améliorer l'accès à l'information. Parmi les méthodes de classification est celle basé sur la collaboration des utilisateurs par l'attribution d'annotation ou tags aux ressources afin de mieux cerner leur contenu.

Cette étude porte sur le développement d'un système de filtrage et de recommandation sur un site de partage de documents scientifiques. Le principe de filtrage des étiquettes est d'abord basé sur leur représentation sous forme d'un graphe pondéré, puis un algorithme de filtrage et de sélection est appliqué pour identifier et recommander les tags les plus importants selon le principe de calcul du plus court chemin sur le graphe de tags.

L'objectif est d'aider l'utilisateur d'un site de partage de publications scientifiques à mieux annoter les documents scientifiques en recommandant une liste de tags significatives.

Mots clés: tagging collaboratif, systèmes de recommandation, filtrage de tags, algorithmes de recommandation.

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABRÉVIATIONS

API	**application programming interface**
UML	**Unified Modeling Language**
HTML	**Hyper Text Manipulation Language**
TF	**Term Frequency**
IDF	**Inverse Document Frequency**
TFP	**Term Frequency Pertinence**
etc	**End of Thinking Capacities**
CF	**Collaborative Filtering**
FC	**Filtrage Collaboratif**
HF	**Hybrid Filtering**
CD	**Compact Disc**
SF	**Social Filtering**
SR	**Système de Recommandation**
CBF	**Content Based Filtering**
NN	**Nearby Neighbors**
IR	**Information retrieval**
RS	**Recommendation system**
IDR	**Integrated Development Environment**
GUI	**Graphical User Interface**

INTRODUCTION GÉNÉRALE

Introduction générale

Face au volume croissant d'informations en ligne, les systèmes de recommandation constituent un moyen efficace pour surmonter le problème de surcharge d'informations. L'utilité des systèmes de recommandation ne peut être surestimée, compte tenu de son adoption généralisée dans de nombreuses applications Web, ainsi que de leur impact potentiel sur l'amélioration de nombreux problèmes liés à un choix excessif.

Alors que l'internet continue à mûrir et devient plus accessible à l'utilisateur, la quantité d'informations disponibles augmente de manière exponentielle. En conséquence, il est de plus en plus difficile de trouver des informations utiles et pertinentes.

En outre, une grande partie de l'information disponible est hautement subjective et est donc difficile à l'évaluer uniquement à l'aide d'algorithmes. De nature subjective, une personne peut aimer absolument quelques choses, tandis que la suivante peut en détester, aucune autorité n'existe. C'est dans ces cas-là que les gens sont extrêmement plus efficaces que les machines pour évaluer et filtrer ces informations. Pour cette raison, l'idée du filtrage collaboratif (FC) a été lancée, et elle fait l'objet de recherches approfondies et a finalement été déployée avec un succès relatif. En utilisant les personnes et les communautés, des informations subjectives peuvent être recommandées par le biais d'appariement d'utilisateurs ou de ressources similaires.

L'avènement récent des systèmes de marquage social (social tagging) et de leurs caractéristiques justifie un autre regard sur les systèmes de FC. Les systèmes de marquage social reposent sur les concepts similaires d'utilisation de la communauté pour trier et organiser les informations: ces systèmes permettent aux utilisateurs d'attacher des tags (mots-clés en langage naturel de leur choix) pour décrire des ressources. Par la suite, ces tags sont utilisés pour la récupération ultérieure et la découverte de ressources, non seulement par l'utilisateur d'origine, mais également par l'ensemble de la communauté d'utilisateurs. Il est intéressant de noter que ces tags sont utilisés à différentes fins, notamment pour indiquer le sujet lui-même, la catégorie ou les caractéristiques d'affinage de la ressource.

Ainsi, les tags semblent fournir le lien manquant dans le FC: elles fournissent le sujet, la catégorie ou un trait de précision d'une ressource, en d'autres termes, le contexte dans lequel l'utilisateur a aimé et par la suite marqué une ressource. Avec ces caractéristiques, les systèmes d'étiquetage social semblent constituer une combinaison bien adaptée d'intégration sociale et de contexte à associer aux systèmes de FC.

Dans ce mémoire de fin d'études intitulée « FILTRAGE DES TAGS DANS UN ENVIRONNEMENT COLLABORATIF », nous proposons un système de recommandation des tags qui consiste à filtrer les tags moins importants à partir d'un graphe pondéré de termes préalablement crée puis à appliquer un algorithme de sélection et de recommandation dans le but de générer des tags pertinents pour un utilisateur cible, dans le but d'aider l'utilisateur d'un site de partage de publications scientifiques à mieux tagger les documents scientifiques.

Ce mémoire est organisé comme suit :

Le chapitre 1 : dans ce chapitre, nous présentons une vue générale sur les systèmes de recommandation ainsi que leurs avantages et inconvénients. Ensuite, nous exposons les systèmes de recommandation et tagging social et à base de graphe. Puis, nous terminons par l'exposition du problème du plus court chemin et nous expliquons l'algorithme de Dijkstra sur la base d'un exemple.

Le chapitre 2 : nous présentons la conception de notre système en utilisant le langage de modélisation UML.

Le chapitre 3 : est consacré à la présentation et l'implémentation du système réalisé. Ensuite, nous discutons les résultats des expérimentations et des évaluations élaborées.

CHAPITRE 1 : SYSTÈMES DE RECOMMANDATION À L'ÈRE DU WEB COLLABORATIF

1. Introduction:

Dans le web collaboratif l'aspect de production, de partage et de tagging de ressources a pris une grande ampleur de liberté ce qui a conduit à l'apparition de différentes morphologies d'écriture de mots sur le web. Ces mots ou tags sont utilisés pour décrire le contenu des ressources partagées.

L'objectif d'un système de recommandation est de générer des recommandations significatives de produits, d'articles ou de tags susceptibles d'intéresser l'utilisateur. Les suggestions de livres sur Amazon ou de films sur Netflix ou de tags sur Citeulike et Flickr sont des exemples concrets du fonctionnement de systèmes de recommandation. La conception de ces moteurs de recommandation dépend du domaine et des caractéristiques particulières des données disponibles.

Dans ce chapitre, nous fournissons un aperçu de la terminologie et des techniques liées aux systèmes de recommandation à l'ère du web collaboratif.

2. Définition d'un système de recommandation :

Il existe une définition générale de Robin Burke [Burke, 2002] qui les définit comme suit :

"*Des systèmes capable de fournir des recommandations personnalisées permettant de guider l'utilisateur vers des ressources intéressantes et utiles au sein d'un espace de données important*".

Le système permet de suggérer à l'utilisateur une liste de recommandations pertinentes qui sont susceptibles de l'intéresser, en apprenant ses préférences à partir de son profil ou l'utilisateur donne ses préférences comme entrée au système pour sélectionner les bonnes recommandations à base de ces derniers.

Les systèmes de recommandation ont prouvé ces dernières années qu'ils sont un bon moyen pour faire face au problème de surcharge cognitive. En effet pour résoudre ce problème, un système de recommandation met en avant des items inconnus qui peuvent être pertinents pour les utilisateurs. Ce niveau de pertinence est déterminé par le système en fonction de connaissances variées (profil de l'utilisateur, contexte de consommation, items disponibles, historique des transactions, feedbacks d'autre utilisateur sur l'item, etc.). L'utilisateur peut alors parcourir les recommandations et peut fournir un feedback implicite ou explicite [27].

Par exemple, dans le site de partage de publications scientifiques Citeulike, lorsqu'un utilisateur fait une recherche à base de tag, le site recommande les tags les plus actives à l'utilisateur dans un nuage de tags, la taille d'un tag reflète sa fréquence d'utilisation: plus la police est grande, plus ce tag est utilisé.

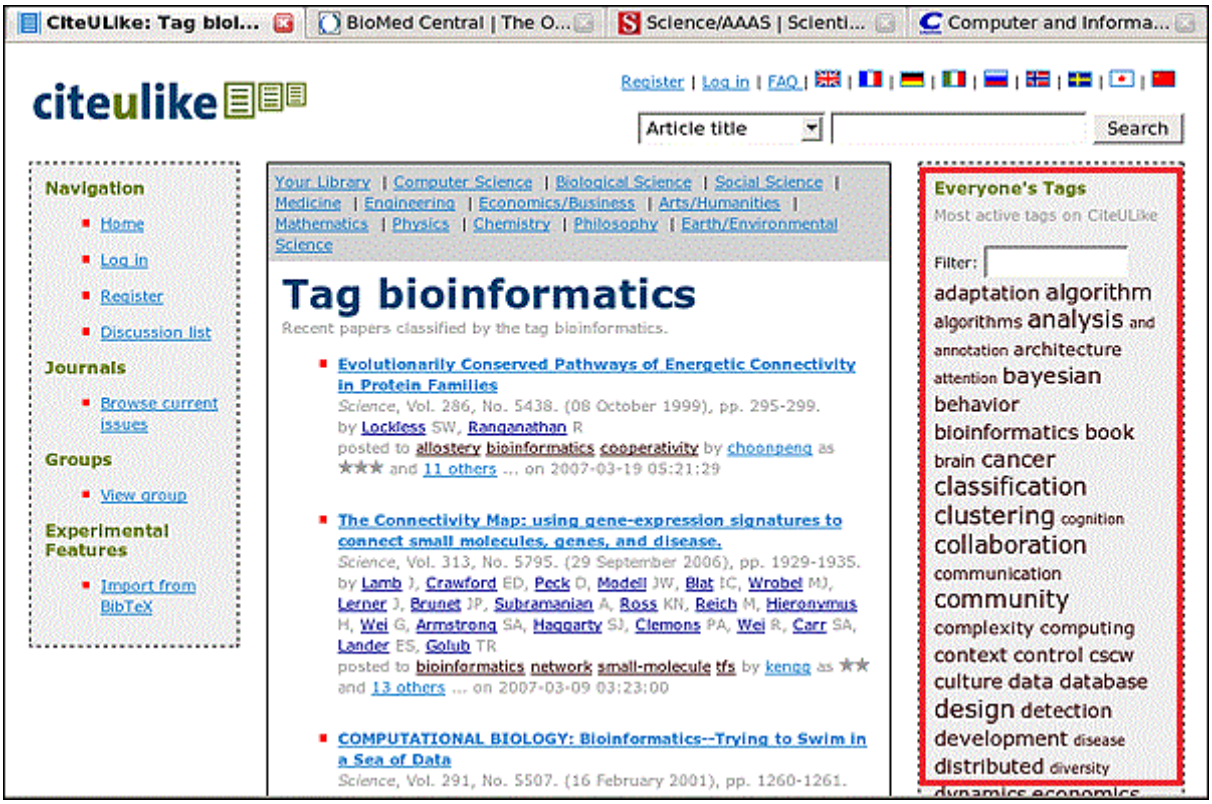


Figure 1.1 : le site Citeulike recommande les tags les plus populaires et plus utilisés lors d'une recherche à base de tag.[50]

3. Filtrage de l'information :

Le filtrage d'informations consiste à effectuer une sélection de ressources (documents, termes,…etc.) intéressantes et d'éliminer celles qui ne le sont pas par rapport à un critère donné pouvant satisfaire les besoins d'utilisateur. Le filtrage offre à l'utilisateur un gain en effort et en temps [46].

4. Types de systèmes de recommandations :

Les systèmes de recommandation (SR) sont une forme spécifique de filtrage de l'information visant à présenter les éléments d'information (musique, livres, news, images, etc.) qui sont susceptibles d'intéresser l'utilisateur [2]. Il y a trois types de systèmes de recommandation sont généralement reconnus en fonction de la manière dont les recommandations sont formulées

[1], à savoir les systèmes de filtrage basé sur le contenu (CBF), de filtrage collaboratif (CF) et de filtrage social (SF). Un système CBF suggère à l'utilisateur des éléments similaires à ceux qu'il préférait ou aimait dans le passé, un système CF propose à l'utilisateur un élément que les personnes ayant les mêmes préférences aimaient auparavant, et un système SF suggère des éléments en fonction des préférences des contacts sociaux de l'utilisateur dans un réseau social. Chacune de ces recommandations a ses forces et ses faiblesses. Afin de remédier aux lacunes particulières et de les compenser, des combinaisons de différentes approches de recommandation sont généralement développées, formant ainsi les systèmes dits de filtrage hybride (HF) [1].

4.1. Les SR basés sur le filtrage collaboratif:

Le filtrage collaboratif est le processus de filtrage ou d'évaluation d'éléments à l'aide de l'opinion d'autres personnes. Bien que le terme filtrage collaboratif (CF) ne soit utilisé que depuis un peu plus d'une décennie, le CF tire ses racines de ce que les humains font depuis des siècles - partager leurs opinions avec d'autres.

Les ordinateurs et le Web nous permettent au lieu de nous limiter à des dizaines, voire des centaines d'individus, Internet nous permet d'examiner les opinions de milliers de personnes. La vitesse des ordinateurs nous permet de traiter ces opinions en temps réel et de déterminer non seulement ce qu'une communauté beaucoup plus large pense d'un élément, mais également de développer une vue vraiment personnalisée de cet élément en utilisant les opinions les plus appropriées pour un utilisateur ou un groupe d'utilisateurs donné [3].

Les systèmes de filtrage collaboratif (CF) fonctionnent en collectant les commentaires des utilisateurs sous forme d'évaluations pour des éléments d'un domaine donné et en exploitant les similarités de comportement d'évaluation entre plusieurs utilisateurs pour déterminer comment recommander un élément.

Les systèmes de filtrage collaboratif produisent des prévisions ou des recommandations pour un utilisateur donné et un ou plusieurs éléments. Les articles peuvent être constitués de tout ce pour quoi un humain peut donner une note, telle que des œuvres d'art, des livres, des CD, des articles de journaux ou des destinations de vacances. Les évaluations dans un système de filtrage collaboratif peuvent revêtir diverses formes.

• Les notations scalaires peuvent consister en des notations numériques, telles que les 1-5 étoiles fournies dans MovieLens, ou des notations ordinales telles que fortement d'accord, d'accord, neutre, en désaccord, tout à fait en désaccord.

• Choix du modèle d'évaluation binaire: d'accord / pas d'accord ou bon / mauvais.

• Les notations unaires peuvent indiquer qu'un utilisateur a observé ou acheté un article, ou d'une autre manière l'a évalué positivement. L'absence de note indique que nous n'avons aucune information concernant l'utilisateur à l'article (peut-être qu'il a acheté l'article ailleurs).

Les notations peuvent être collectées par des moyens explicites, implicites ou les deux. Les évaluations explicites sont celles dans lesquelles un utilisateur est invité à donner son avis sur un élément. Les évaluations implicites sont celles déduites des actions d'un utilisateur. Par exemple, un utilisateur qui visite une page de produit a peut-être un intérêt pour ce produit, tandis qu'un utilisateur qui achète par la suite le produit peut avoir un intérêt beaucoup plus fort pour ce produit [4].

En général, les approches des FC sont généralement classées en deux catégories principales: basée sur un modèle et basée sur la mémoire.

4.1.1. Filtrage collaboratif basée sur un modèle:

Les approches basées sur des modèles construisent des modèles statistiques de modèles d'évaluation utilisateur / élément afin de fournir des prédictions d'évaluation automatiques.

Les algorithmes de filtrage collaboratif basés sur un modèle fournissent des recommandations d'éléments en développant d'abord un modèle d'évaluation d'audience.

Les algorithmes de cette catégorie adoptent une approche probabiliste et envisagent le processus de filtrage collaboratif comme le calcul de la valeur attendue d'une prévision de l'utilisateur, compte tenu de ses évaluations d'autres éléments. Le processus de création de modèle est effectué par différents algorithmes d'apprentissage machine tels que le réseau bayésien, la mise en cluster et les approches basées sur des règles.

Le modèle de réseau bayésien [5] formule un modèle probabiliste pour le problème du filtrage collaboratif. Le modèle de classification traite le filtrage collaboratif comme un problème de classification [6] et fonctionne en regroupant des utilisateurs similaires dans la même classe et en estimant la probabilité qu'un utilisateur particulier se trouve dans une classe particulière C, puis à partir de là, calcule la probabilité conditionnelle d'évaluation.

L'approche basée sur des règles applique des algorithmes de découverte de règles d'association pour rechercher une association entre des articles co-achetés, puis génère une recommandation d'élément basée sur la force de l'association entre les éléments [7].

4.1.2. Filtrage collaboratif basée sur la mémoire:

Les approches basées sur la mémoire, en d'autre part, font des prédictions d'évaluation basées sur l'ensemble de la collection d'évaluations [8]. Ces approches peuvent être des stratégies basées sur les utilisateurs et les éléments.

Les stratégies **basées sur les utilisateurs** reposent sur le principe que les enregistrements d'évaluation d'un utilisateur particulier ne sont pas également utiles à tous les autres utilisateurs en tant qu'éléments permettant de fournir des suggestions d'éléments personnels [9].

Les aspects centraux de ces algorithmes sont donc :

a) comment identifier les voisins qui constituent la meilleure base pour générer des recommandations d'articles pour l'utilisateur cible.

b) comment utiliser correctement les informations fournies par eux.

En règle générale, l'identification de voisinage est basée sur la sélection des utilisateurs qui sont plus similaires à l'utilisateur cible en fonction d'une métrique de similarité [8].

La similarité entre deux utilisateurs est généralement calculée a) en recherchant un ensemble d'articles avec lesquels les deux utilisateurs ont interagi et b) en examinant dans quelle mesure les utilisateurs affichaient des comportements similaires (par exemple, modèles de navigation, de navigation et d'achat) pour ces éléments. Cette approche de base peut être complétée par des comparaisons alternatives de pratiquement toutes les fonctionnalités utilisateur auxquelles un système a accès, telles que les données démographiques et de réseau social personnelles. Il est également courant de définir un nombre maximal de voisins (ou un seuil minimal de similarité) afin de limiter la taille du quartier, soit pour des performances informatiques optimales, soit pour éviter les utilisateurs bruyants qui ne sont pas assez similaires. Une fois que les voisins de l'utilisateur cible sont sélectionnés, plus un voisin est semblable à l'utilisateur, plus ses préférences sont prises en compte en tant qu'entrée pour produire des recommandations. Par exemple, une approche commune basée sur l'utilisateur consiste à prédire la pertinence d'un élément pour l'utilisateur cible par une combinaison

linéaire des notations de ses voisins, pondérées par la similarité entre l'utilisateur cible et ces voisins [10].

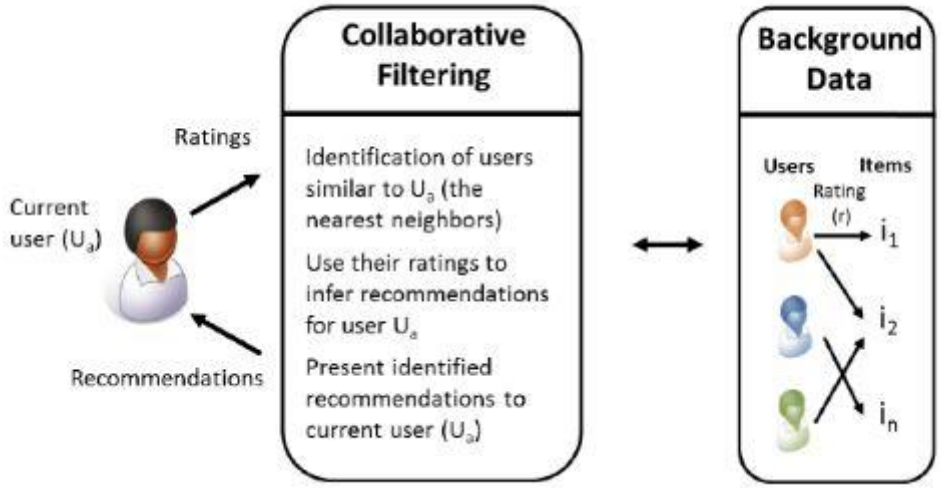


***Figure 1.2*:** Filtrage collaboratif (CF): les utilisateurs évaluent les éléments et reçoivent des recommandations pour les éléments en fonction des évaluations des utilisateurs ayant un comportement d'évaluation similaire - les voisins les plus proches (NN).

Les stratégies **basées sur les objets** quant à elles, reconnaissent des modèles de similitude entre les éléments eux-mêmes, plutôt qu'entre les choix de l'utilisateur, comme le font les approches basées sur l'utilisateur. En général, les recommandations basées sur les éléments examinent chaque élément de la liste des éléments choisis / évalués de l'utilisateur cible, et trouvent d'autres éléments qui semblent être «similaires» à cet élément [11]. La similarité d'éléments est généralement définie en termes de corrélation de notation entre utilisateurs, bien que des similitudes basées sur le cosinus ou sur la probabilité aient également été proposées.

4.2. Les SR basés sur le contenu :

Les systèmes mettant en œuvre une approche de recommandation basée sur le contenu analysent un ensemble de documents et / ou des descriptions d'éléments précédemment notés par un utilisateur et construisent un modèle ou un profil des intérêts des utilisateurs en fonction des caractéristiques des objets évalués par cet utilisateur [12].

Le profil est une représentation structurée des intérêts des utilisateurs, adoptée pour recommander de nouveaux éléments intéressants. Le processus de recommandation consiste

essentiellement à faire correspondre les attributs du profil utilisateur avec les attributs d'un objet de contenu. Le résultat est un jugement de pertinence qui représente le niveau d'intérêt de l'utilisateur pour cet objet.

Si un profil reflète avec précision les préférences de l'utilisateur, il est extrêmement avantageux pour l'efficacité d'un processus d'accès à l'information. Par exemple, il pourrait être utilisé pour filtrer les résultats de la recherche en déterminant si un utilisateur est intéressé par une page Web spécifique ou non et, dans le cas contraire, en l'empêchant de s'afficher.

Par exemple, étant donné les informations sur le genre de film et sachant qu'un utilisateur aime «Spiderman» et «Black Panther», on peut en déduire une prédilection pour la science-fiction et donc recommander «Avengers: infinity war». Les recommandeurs basés sur le contenu font référence à de telles approches, qui fournissent des recommandations en comparant les représentations de contenu décrivant un élément à des représentations de contenu qui intéressent l'utilisateur. Ces approches sont parfois également appelées filtrage par contenu.

Beaucoup de recherches dans ce domaine ont été consacrées à la recommandation d'articles comportant un contenu textuel associé, tels que des pages Web, des livres et des films; où les pages Web elles-mêmes ou le contenu associé, comme les descriptions et les commentaires des utilisateurs, sont disponibles.

Ainsi, plusieurs approches ont traité ce problème comme une tâche de récupération d'informations (IR), dans laquelle le contenu associé aux préférences de l'utilisateur est traité comme une requête et où les documents non notés sont notés avec pertinence / similarité avec cette requête [13].

Une alternative aux approches IR consiste à traiter les recommandations comme une tâche de classification, chaque exemple représentant le contenu d'un élément, et les évaluations passées d'un utilisateur étant utilisées comme étiquettes pour ces exemples. Dans le domaine des recommandations de livres, un texte de champs tels que le titre, l'auteur, les synopsis, les critiques et les termes du sujet est utilisé pour former un classifieur multinomial naiveBayes. Les classements sur une échelle de 1 à k peuvent être directement mappés sur k classes [14], ou bien le classement numérique peut être utilisé pour pondérer l'exemple de formation dans un paramètre de classification binaire probabiliste [15].

D'autres algorithmes de classification ont également été utilisés pour des recommandations purement basées sur le contenu, notamment les k-voisins les plus proches, les arbres de décision et les réseaux de neurones [16].

4.3. Les SR hybrides :

La prolifération de nouvelles stratégies de recommandation donne lieu à une variété croissante d'options disponibles pour le développement de systèmes de recommandation. La recherche en apprentissage automatique montre depuis longtemps que la combinaison de méthodes donne généralement de meilleurs résultats que chaque méthode séparément, ce qui est également le cas dans RecommenderSystems - le prix Netflix en est un exemple paradigmatique, où toutes les équipes classées de haut niveau utilisaient de grands ensembles de recommandation, ce qui peut être considéré comme un cas d'approches de filtrage hybride.

Dans une telle approche hybride, la décision la plus importante est de combiner les informations. Cependant, il faut d'abord décider du type d'information qui sera utilisé dans l'ensemble. L'approche standard dans la littérature est de combiner les recommandations de la CBF et des CF, en surmontant les problèmes de parcimonie et de fonctionnalités restreintes de chaque recommandation.

Les recommandeurs hybrides combinent des aspects de différents algorithmes de recommandation dans le but de créer un algorithme de recommandation qui peut exploiter les points forts de ses algorithmes de composants tout en atténuant leurs faiblesses.

Plusieurs algorithmes de recommandation hybrides incorporant des balises d'une ou plusieurs manières ont été proposés [30].

Dans [17], une taxonomie détaillée des systèmes de recommandation hybrides est présentée, classant les approches existantes dans les types suivants:

- **En cascade**: la recommandation est effectuée comme un processus séquentiel de manière à ce qu'un des recommandateurs affine les recommandations données par l'autre.
- **Augmentation des fonctionnalités**: la sortie d'un programme de recommandation est utilisée en tant que fonction d'entrée supplémentaire pour un autre groupe de recommandation.

- **Combinaison de fonctionnalités**: les fonctionnalités utilisées par différents recommandateurs sont intégrées et combinées dans une source de données unique, exploitée par un seul recommandateur.
- **Méta-niveau**: le modèle généré par l'un des recommandateurs est utilisé comme entrée pour un autre recommandateur. Comme indiqué dans [17]: «Cela diffère de l'augmentation des caractéristiques: dans un hybride de l'augmentation, nous utilisons un modèle appris pour générer des caractéristiques à saisir dans un deuxième algorithme; dans un hybride de méta-niveau, le modèle entier devient l'entrée. "
- **Mixte**: les recommandations de plusieurs recommandateurs sont disponibles et sont présentées ensemble en même temps au moyen de certaines stratégies de classement ou de combinaison.
- **Pondéré**: les scores fournis par les recommandateurs sont agrégés à l'aide d'une combinaison linéaire ou d'un schéma de vote.
- **Commutation**: cas particulier du type précédent prenant en compte les poids binaires, de telle sorte que l'un des recommandateurs soit activé et les autres désactivés.

L'utilisation d'un type spécifique de méthode de recommandation hybride dépend de l'application finale, mais, plus important encore, du type de recommandations combinées.

5. Propriétés des systèmes de recommandations:

Dans le contexte de la recommandation, les chercheurs et les professionnels des RS sont soucieux de la satisfaction des utilisateurs, de sorte que les prévisions puissent offrir davantage de valeur à l'utilisateur. La raison en est que les RS doivent être utiles à l'utilisateur, en ne leur suggérant pas seulement de consommer «plus de la même chose». Les chercheurs s'inquiètent de l'interaction des utilisateurs et de l'expérience de consommation dans le système. Récemment, des chercheurs ont tenté de résoudre ce problème en évaluant différents concepts d'évaluation, au lieu d'utiliser simplement des techniques d'exactitude prédictive et d'apprentissage automatique. La performance des suggestions fournies par un RS doit être mesurée par la valeur qu'il peut générer pour l'utilisateur. Il existe de nombreux concepts concernant l'évaluation des recommandations, tels que la couverture, la nouveauté, la diversité et les surprises des recommandations, qui ont été évalués par différentes recherches [18].

Jetons un coup d'œil aux différentes fonctionnalités des systèmes de recommandation :

- **Couverture**: Mesure dans laquelle les recommandations couvrent tous les éléments et actions disponibles.
- **Confiance**: le système peut dire à l'utilisateur à quel point il fait confiance à ses recommandations et combien d'utilisateurs font confiance au système. Souvent, si un système explique ses recommandations et qu'elles sont raisonnables, l'utilisateur fait confiance au système.
- **Démarrage à froid**: lorsque le système recommande des éléments que les utilisateurs ne connaissaient pas auparavant.
- **Sérendipité**: quelle surprise l'utilisateur trouve les recommandations.
- **Diversité**: plus les recommandations sont diverses, plus l'offre aux utilisateurs est large.
- **Utilitaire**: mesure de l'utilité de la recommandation pour l'utilisateur.
- **Risque**: lorsque le système de recommandation peut indiquer à l'utilisateur le risque de suivre une recommandation.
- **Robustesse**: la stabilité du système face aux abus ou aux fausses informations.
- Confidentialité: bien que les utilisateurs donnent volontiers des informations aux systèmes de recommandation, ils souhaitent toujours que ces informations soient privées.
- **Adaptivité**: lorsque le système de recommandation peut adapter et fournir aux utilisateurs des recommandations pertinentes, même lorsque le contenu et l'environnement sont dynamiques.
- **Évolutivité**: le système peut gérer une quantité croissante de données.

La plupart des concepts existants peuvent être résumés en 5 concepts différents: utilité, Démarrage à froid, diversité, sérendipité et couverture.

Nous étudions les principales définitions et mesures pour chacune d'elles.

5.1. L'utilité:

L'utilité a été mentionnée dans la littérature sous de nombreux noms, tels que pertinence, utilité, valeur de recommandation et satisfaction. Dans le Manuel des systèmes de recommandation, Ricci et al. [29] soutient que cet utilitaire représente la valeur que l'utilisateur reçoit en étant recommandé. Comme leur propre définition l'indique, si l'utilisateur apprécie les éléments recommandés, il / elle reçoit des recommandations utiles. De plus, l'utilité a été définie comme un ordre de préférence de consommation. Si les

utilisateurs ne consommaient que ce qu'ils préféraient en premier lieu, par conséquent, recommander de tels produits l'aiderait à les trouver facilement, ce qui serait utile pour la recommandation Herlocker et al. [18].

Comme on peut le constater, la plupart des définitions associent utilitaire aux désirs de consommation de l'utilisateur et au fait que celui-ci ait apprécié les recommandations. Dans cette définition, les paramètres permettant d'évaluer l'utilité de la recommandation devraient être centrés sur la manière dont l'utilisateur pourrait réagir aux prédictions de l'auteur de la recommandation. Ricci et al. [19] mentionnent que l'utilité pourrait être mesurée en évaluant la note attribuée par l'utilisateur aux éléments prédits après les avoir consommés. Cette méthode est susceptible d'être correcte et de capturer si les recommandations apportent une valeur à l'utilisateur, mais cela impliquerait une évaluation en ligne coûteuse.

5.2. Démarrage à froid:

Les nouveaux éléments et les nouveaux utilisateurs représentent un défi important pour les systèmes de recommandation. Ensemble, ces problèmes sont appelés le problème du démarrage à froid [20]. Le premier de ces problèmes se pose dans les systèmes de filtrage collaboratif, dans lesquels un élément ne peut être recommandé à moins qu'un utilisateur ne l'ait évalué auparavant. Cette question concerne non seulement les nouveaux éléments, mais également les éléments obscurs, ce qui est particulièrement préjudiciable pour les utilisateurs aux goûts éclectiques. En tant que tel, le problème des nouveaux articles est également souvent appelé le problème du premier évaluateur. Puisque les approches basées sur le contenu [21] ne reposent pas sur des évaluations d'autres utilisateurs, elles peuvent être utilisées pour produire des recommandations pour tous les éléments, à condition que leurs attributs soient disponibles. En fait, les prédictions basées sur le contenu d'utilisateurs similaires peuvent également être utilisées pour améliorer encore les prédictions pour l'utilisateur actif [22].

Le problème des nouveaux utilisateurs est difficile à résoudre car, sans les préférences préalables d'un utilisateur, il n'est pas possible de trouver des utilisateurs similaires ni de créer un profil basé sur le contenu. En tant que telles, les recherches dans ce domaine ont principalement porté sur la sélection efficace des éléments à évaluer par un utilisateur, de manière à améliorer rapidement les performances de la recommandation avec le moins de commentaires de l'utilisateur. Dans ce contexte, les techniques classiques d'apprentissage actif peuvent être exploitées pour résoudre le problème de la sélection d'éléments [23].

5.3. Diversité:

La diversité est un concept qui concerne la diversité des éléments de la liste de recommandations. Il a également été largement étudié par les chercheurs précédents.

D'après Ricci et al. [19], la diversité des RS a l'effet contraire de la similitude. Les auteurs indiquent que les listes de recommandations peu variées peuvent ne pas intéresser l'utilisateur.

À la suite de cette définition, les mesures proposées tendent à calculer la diversité comme une dissimilarité entre les éléments de la liste de recommandations. Ziegler et al. [24] ont proposé une métrique pour la similarité intra-liste.

La fonction d (i, j) calcule la distance entre les éléments i et j dans la liste de recommandations R_u . Cette métrique capture en réalité la similarité de la liste; par conséquent, les valeurs basses pour cette métrique représentent une liste plus similaire, dans laquelle les éléments sont similaires les uns aux autres.

$$(R_u) = \sum_{i \in R_u} \sum_{j \in R_u i \neq j} d(i,j) \tag{1.1}$$

5.4. Sérendipité:

Le terme sérendipité signifie une trouvaille ou une surprise satisfaisante. D'après Ricci et al. [19], la sérendipité représente des recommandations surprenantes.

Les auteurs utilisent le mot sérendipité comme concept d'objet surprenant et intéressant pour l'utilisateur.

En termes simples, la plupart des utilisateurs n'évaluent pas la plupart des éléments et, par conséquent, la matrice des évaluations des utilisateurs est généralement très rare. Ceci est un problème pour les systèmes de filtrage collaboratif, car il diminue la probabilité de trouver un ensemble d'utilisateurs avec des évaluations similaires.

Ce problème se produit souvent lorsqu'un système présente un ratio élément / utilisateur très élevé ou lorsqu'il est au stade initial d'utilisation. Ce problème peut être atténué en utilisant des informations de domaine supplémentaires ou en faisant des hypothèses sur le processus de génération de données permettant une imputation de haute qualité [25].

5.5. Couverture:

Selon [26], la couverture d'un programme de recommandation est une mesure du domaine d'éléments sur lequel le système peut faire des recommandations. Dans la littérature, le terme "couverture" était principalement associé à deux concepts: (1) le pourcentage d'éléments pour lesquels le système est capable de générer une recommandation, et (2) le pourcentage d'éléments disponibles qui sont effectivement recommandés à un utilisateur [26].

Bien que différents auteurs diffèrent en ce qui concerne la terminologie, nous adoptons ici la définition de [26] et désignons (1) couverture de prédiction et (2) couverture de catalogue. La couverture de prévision dépend fortement du moteur de recommandation et de ses entrées. Si nous considérons le filtrage collaboratif (CF), les entrées sont des évaluations d'élément. Si nous considérons les recommandateurs basés sur la connaissance, les entrées sont une sorte de logique de fin de moyen telle que des règles de recommandation explicites et les préférences de l'utilisateur. Dans les deux cas, le système pourra générer des recommandations pour les articles pour lesquels il a reçu suffisamment d'informations.

6. Systèmes de recommandations et le tagging social :

Les systèmes de recommandation à base de tags permettent de limiter l'effet des problèmes de l'accroissement de la quantité de l'information et d'accès aux ressources gérées. En effet, ces systèmes proposent à l'utilisateur les tags les plus pertinents pour annoter ses ressources [28].
Ils peuvent intervenir au niveau de deux tâches :
-La tâche d'indexation en aidant l'utilisateur à sélectionner les tags les plus appropriés pour annoter ses ressources ;
- La tâche de recherche d'information au sein des répertoires en faisant une recherche par tags [28].

La recommandation de tags consiste à recommander des tags intéressants à un utilisateur sur la base d'un enregistrement de ses préférences antérieures. C'est sans doute la tâche de recommandation la plus populaire et de nombreuses classes différentes d'algorithmes ont été proposées.
Officiellement, le but de chacun des algorithmes de recommandation de tags abordés dans cette section est la recommandation du top-N tags, à cette fin, pour chaque élément représentant la probabilité que ce tag soit pertinent pour l'utilisateur actif. Les

recommandations finales pour un utilisateur sont générées en classant toutes les étiquettes en fonction de leur score prévu [30].

7. Systèmes de recommandation à base de graphe :

Récemment, les méthodes de graphes ont été appliquées aux systèmes de recommandation. Certains chercheurs ont mis en avant l'application de graphes pour apprendre la corrélation entre les utilisateurs ou les éléments. Avec les vecteurs d'utilisateur ou d'éléments mappés sur le réseau, certaines relations latentes entre eux peuvent être découvertes. Par exemple, Shameem Ahamed Puthiya Parambath et al. Ont mis en avant une méthode appliquée aux systèmes de recommandation basée sur un graphe de similarité [33]. Li Xin et al. ont proposé une méthode basée sur le noyau graphique pour apprendre la relation entre les utilisateurs et les éléments [34]. Yuan Zhang et al. Ont proposé une méthode basée sur les graphes et l'étiquette s'appliquant aux systèmes de recommandation [35].

Les résultats expérimentaux réussis de ces méthodes montrent que les FC à base de graphe présentent certains avantages par rapport aux méthodes existantes, car elles peuvent apprendre davantage d'informations latentes sur le réseau.

Cependant, dans toutes ces approches, les éléments (ou le contenu) à recommander apparaissent en tant que nœuds dans le graphe. Nous appelons donc ces approches systèmes de recommandation basés sur des graphes d'items.

Les liaisons, par contre, représentent la similarité entre les éléments. Ces similitudes s'apprennent de l'attitude que différents utilisateurs ont vis-à-vis de différents éléments.

Par exemple, si tous les utilisateurs qui ont aimé l'élément a, ont également aimé l'élément b, il est probable qu'il y aura une liaison (arêtes) entre a et b. De plus, ces arêtes sont pondérées proportionnellement à la similitude entre les deux éléments. Dans certains systèmes de recommandation, chaque utilisateur a son propre graphe, en fonction des éléments que l'utilisateur a appréciés (ou des éléments appréciés par les utilisateurs similaires à l'utilisateur actuel). Encore une fois, ce groupe de systèmes de recommandation basés sur un graphe d'articles requiert des données de nombreux utilisateurs afin de fournir une recommandation à un utilisateur spécifique.

Certaines œuvres utilisent un graphe bipartite dans lequel un ensemble de nœuds représentent des éléments, tandis que le second ensemble de nœuds représente les utilisateurs. Les arêtes

représentent les utilisateurs qui aiment ou achètent des articles spécifiques. Par exemple, si l'utilisateur A a acheté les éléments a, b et c, il existe des arêtes correspondantes dans le graphe de A à a, b et c. Différents travaux utilisent les graphes dans différentes méthodes, dans le but de trouver des éléments du graphe qui devraient être recommandés à l'utilisateur.

Certains travaux utilisent un troisième type de nœuds, tels que des concepts, qui permettent des connexions supplémentaires entre certains éléments et pas nécessairement via des nœuds représentant des utilisateurs. Toutes ces approches basées sur des graphes d'items reposent en réalité sur la collaboration et nécessitent donc des données de nombreux utilisateurs pour obtenir de bons résultats. Ces méthodes ne conviennent pas non plus aux utilisateurs qui sont très sensibles à la confidentialité, car les informations de certains utilisateurs doivent être utilisées afin de recommander des éléments à d'autres utilisateurs [36] [37] [38].

7.1. Problème du plus court chemin :

Un problème qui consiste à rechercher le plus court chemin d'un sommet à un autre en minimisant la somme des poids des arcs.

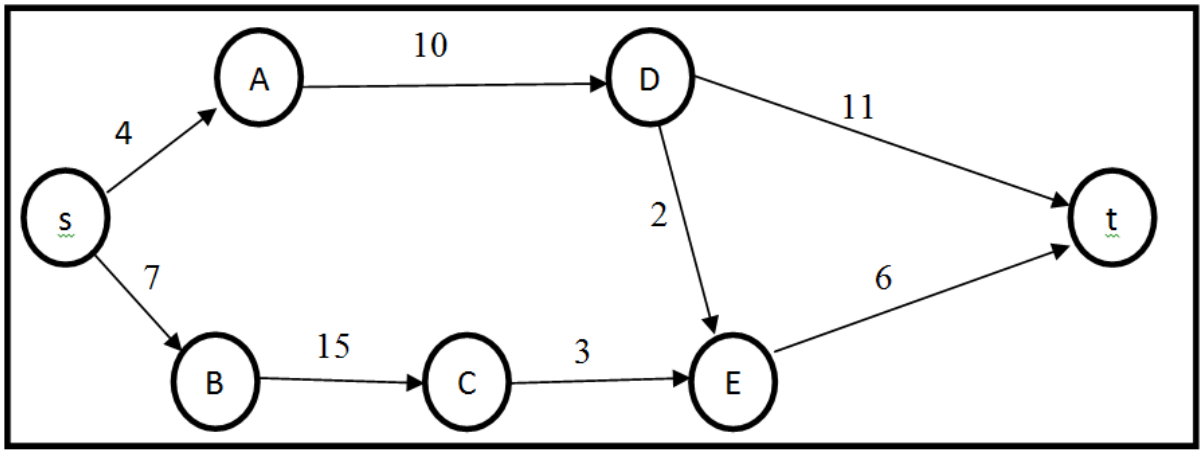
Pour plus d'explication, dans un graphe e G = (V, A, c), un chemin p est défini par un ensemble ordonné de nœuds p = (v_1, v_2, ...,v_l) avec p ⊂ V et l = |p| la longueur du chemin p. Pour chaque couple de nœuds consécutifs i et j du chemin p, il existe un arc (i, j) reliant ces deux nœuds, chaque arc a un poids différent de l'autre.

Dans le problème de recherche du plus court chemin, le premier nœud v_1 appelé source est souvent noté *s*, pour *start*, et le dernier nœud v_l est appelé puits et noté *t* pour *target*.

Le plus court chemin p entre s et t dans le graphe G est celui qui relie les nœuds en minimisant la somme des coûts des arcs c(p) [31].

$$C(p) = \sum_{i+1}^{l-1} c_{v_i,v_{i+1}} \qquad (1.2)$$

Par exemple dans le graphe de figure 1.3 ci-dessous:



***Figue 1.3* :** Chemins dans un graphe

On a plusieurs chemins reliant le nœud s au nœud t ayant différents coûts comme suit :

- p1 = (s, A, D, t) de coûts C (p1) = 25
- p2 = (s, A, D, E, t) de coûts C (p1) = 22
- p3 = (s, B, C, E, t) de coûts C (p1) = 31

Donc p2 est le plus court chemin parce que la somme de ces arcs est la plus minimal que les autres chemins dans le graphe.

Parmi les algorithmes les plus connus qui résout le problème du plus court chemin :

7.2. Algorithme de Dijkstra :

L'algorithme de Dijkstra résout un problème plus général. Il permet de calculer le plus court chemin d'un nœud vers tous les autres nœuds du graphe. Son principe est présenté dans l'algorithme 1. Une particularité de cet algorithme est qu'il ne fonctionne que sur les graphes valués avec des coûts positifs c'est-à-dire avec une fonction de coût de type c : A → R + qui associe à chaque arc un coût positif ou nul [31].

Algorithme de Dijkstra

1: Q ← V // la file Q contient tous les nœuds à traiter
2: $distance_i = +\infty$, i = {1, ..., n} \ {s}
3: $arc_i = \varphi$, i = {1, ..., n}
4: $distance_s = 0$
5: **tant que** Q ≠ φ **faire** // tant qu'il reste des nœuds à traiter
6: i ← Q.Min() // sélectionner le nœud de plus petite distance
7: Q ← Q \ i
8: **pour tout** (i, j) ∈ A **faire** // Pour chaque arc sortant du nœud i
9: **si** $distance_i + c_{ij} < distance_j$ **alors**

10: $distance_j \leftarrow distance_i + c_{ij}$
11: $arc_j \leftarrow (i, j)$
12: **fin si**
13: **fin pour**
14: **fin tant que**

L'étiquette $distance_i$ est associée à chaque nœud i ∈ V et contient la valeur de coût minimal correspondant au plus court chemin entre le nœud source s et le nœud i. au début de l'algorithme, tous les étiquettes sont initialisées à +∞ sauf le nœud s (nœud de Départ) pour lequel l'étiquette est nulle 0.

Q contient tous les nœuds de V. Au cours de chaque itération, retirer les nœuds un à un de la file Q, on prendre toujours le nœud i de plus petite étiquette$distance_i$ (ligne 6), Ensuite, on met à jour les étiquettes des nœuds successeurs et la nouvelle étiquette du nœud est le minimum entre l'étiquette existante et celle obtenue en ajoutant le poids de l'arc entre nœud successeurs et le nœud i à la distance du le nœud i.

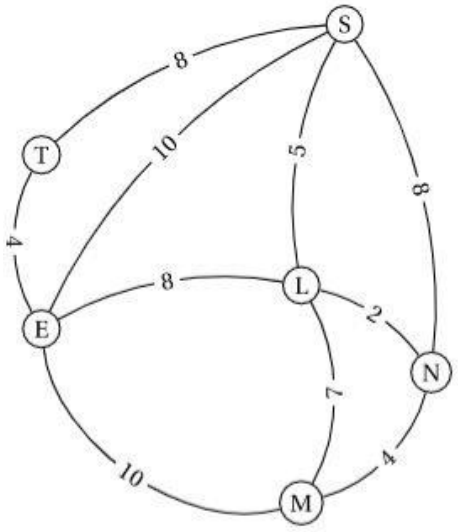
On continue ainsi jusqu'à ce qu'il ne reste plus de nœuds à traiter.

La variable arc_j d'un nœud j permet de garder en mémoire l'arc précédent (i, j) de manière àpouvoir reconstruire le chemin complet pour arriver de s à j.

L'algorithme de Dijkstra calcule les plus courts chemins d'un nœud source s vers tous les autres nœuds, ce qui correspond à l'arbre des plus courts chemins. Cependant, si l'objectif initial est de calculer uniquement le plus court chemin entre deux nœuds, comme de s vers t, il est possible d'arrêter l'algorithme dès que le nœud t est retiré de la file Q. En effet, comme il s'agit d'un algorithme de label-setting, lorsque t est retiré de la file Q, nous sommes alors certains que son étiquette associée correspond au coût du plus court chemin de s à t.

La complexité de l'algorithme de Dijkstra est en O (n^2). Il peut-être implémenté en O (m + n log(n)) avec des monceaux de Fibonnacci [31].

Par exemple on a un graphe de figure 1.4 avec le résultat de l'exécution de l'algorithme de Dijkstra avec cinq étapes ci-dessous :



	E	L	M	N	S	T
Départ	∞	∞	0_M	∞	∞	∞
M (0)	10_M	7_M		4_M	∞	∞
N (4)	10_M	6_N			12_N	∞
L (6)	10_M				11_L	∞
E (10)					11_L	14_E

Figure 1.4 **:** Exemple d'exécution de l'algorithme de Dijkstra

Comme on a vue dans la figure 1.4 on va rechercher le chemin le plus court menant de M à S, toutes les étiquettes sont initialisées à +∞ sauf le nœud de Départ à 0.

À l'étape 1 (départ), on sélectionne toujours **le plus petit résultat,** c'est «0_M» qui correspond au chemin menant au **sommet M en 0,**à partir de M, on voit sur le graphe que l'on peut rejoindre E, L et N en respectivement 10, 7 et 4.

À l'étape 2 (M(0)), On sélectionne **le plus petit résultat** de la dernière ligne. Ici, c'est «4_M» qui correspond au chemin menant au **sommet N** en 4.

En ajoutant le poids de l'arc entre M et N (arc (M, N)=4) à la distance du le nœud M alors la somme est égale à 4.

À partir de N, on peut rejoindre L et S respectivement avec l'ajout de distance de nœud i 6, 12(on ne se préoccupe plus de M qui a été « désactivé »).

Et on refait sa jusqu'à arriver à la dernière étape, on arrive alors au nœud S avec distance égale à 11, on trouve le plus court chemin à partir du tableau comme suit :

- On recherche la cellule marquée en rouge de la colonne S, elle contient 11_L. On note la lettre écrite en indice : L.

- On recherche la cellule marquée en rouge de la colonne L, elle contient 6_N. On note la lettre écrite en indice : N.
- On recherche la cellule marquée en rouge de la colonne N, elle contient 4_M . On note la lettre écrite en indice : M.

Alors le plus court chemin est : p = (M, N, L, S) de coûts C (p) = 11.

En domaine des systèmes de recommandation l'algorithme a été utilisé dans une tentative de trouver les nœuds les plus proches du graphe par rapport à tous les nœuds initiaux (Les nœuds initiaux représentent l'intérêt de l'utilisateur), qui sont ensuite renvoyés sous forme de recommandations. Il existe de nombreuses façons d'examiner le problème de la recherche des nœuds les plus proches, en particulier dans un graphe très complexe. Et le principe de l'algorithme de Dijkstra peut résoudre ce problème mais le même algorithme avec des paramètres différents peut être utilisé pour la recommandation [39]. Dans notre travail nous avons essayé d'adapter cet algorithme à notre problème afin de recommander des tags pertinents à partir de notre graphe de termes. Ces tags sont recommandés pour annoter les publications scientifiques sur CiteULike et faciliter ainsi l'accès à ces articles.

8. Conclusion:

Les systèmes de recommandation sont une nouvelle technique puissante et deviennent rapidement un outil crucial du commerce électronique sur le Web.

Dans ce chapitre, nous avons présenté la définition du système de recommandation ainsi que les trois types de ce dernier et les problèmes qu'ils essayent de résoudre. On a vu également la nécessité d'avoir recours à des systèmes de recommandation qui doivent vérifier certaine propriétés comme le cold star, la couverture ou scalabilité. Nous avons présenté finalement l'algorithme de Dijkstra qui résout le problème de plus court chemin. Cet algorithme sera adapté à notre problème de recommandation de tags pour l'annotation de publications scientifiques.

CHAPITRE 2 : CONCEPTION

1. Introduction :

Dans ce chapitre nous allons décrire les grandes étapes de la réalisation de notre système de recommandation de tags. Le chapitre commence tout d'abord par une présentation de l'architecture générale du système suivie d'une description de notre ensemble de donnés qui s'agit d'un graphe pondéré de termes avant de présenter en détail les différentes étapes de travail à savoir les prétraitements faits sur notre base de test le calcul des poids des termes de cette base. Le chapitre présente également l'algorithme de recherche du plus court chemin après l'avoir adapté à notre cas. Quelques diagrammes UML sont aussi élaborés afin de modéliser le système.

2. Architecture générale de notre système :

La réalisation de notre système a passé par trois étapes principales :

- Première étape, nous commençons par l'ajout de nouveaux documents à la base de test. Ces documents s'agissent d'articles scientifiques représentés par leurs titres et leurs résumés seulement.
- Deuxième étape, nous faisons un prétraitement sur les articles de test dont le but est d'extraire les concepts clés et calculer la fréquence de chaque concept.
- Troisième étape, nous appliquons l'algorithme de recommandation sur un article pour recommander des tags significatifs pouvant aider à mieux cerner le contenu restreint de l'article. Les tags sont sectionnés à partir du graphe après avoir appliqué l'algorithme de Dijkstra adapté. Les tags suggérés sont évalués pour mesurer leur significativité.

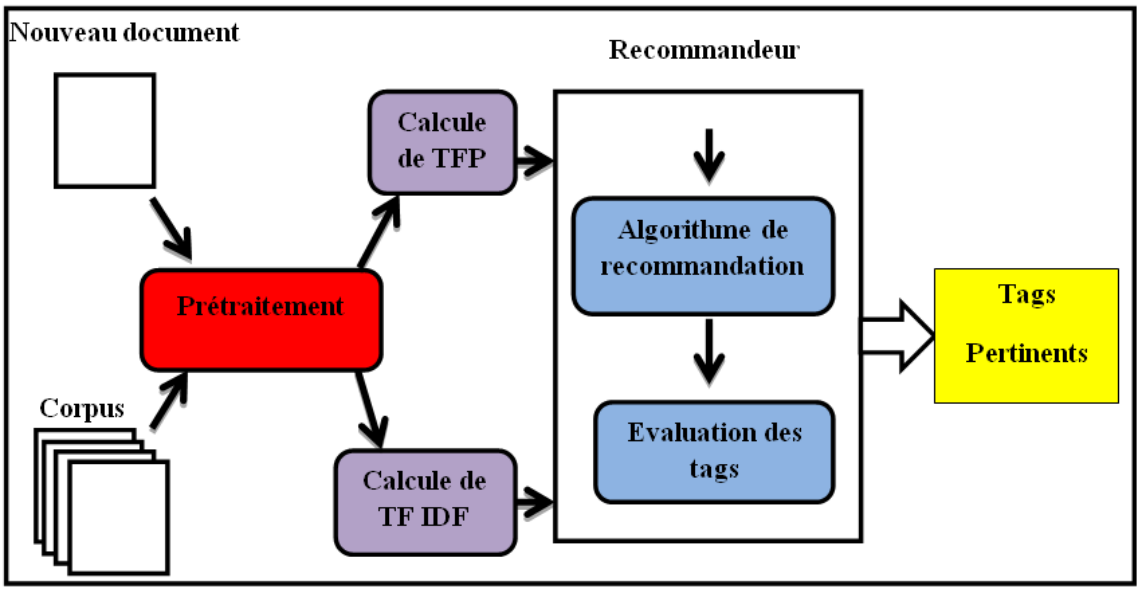


***Figure 2.1*:** Architecture général de système

2.1.Présentation de notre graphe (base de données) :

Nous considérons le graphe ci-dessous notre base de données, ce graphe est construit à partir d'une collection d'articles. Chaque nœud dans ce graphe comporte un terme (qui est soit un tag ou un concept) les termes sont reliés par des liaisons pondérés. Les poids des liaisons représentent le degré de similarité entre deux termes. Le graphe comporte 20101 nœuds.

Les étapes de création du graphe ont été abordées en détail dans le travail de [44].

La figure (2.2) représente le graphe, elle est prise à l'aide d'un logiciel de visualisation (Gephi). Gephi est un logiciel open source de visualisation et d'analyse de réseau, écrit en Java. Il aide les analystes de données à révéler intuitivement les modèles et les tendances, et à mettre en évidence les valeurs aberrantes. [49]

Dans le reste du mémoire, nous appelons par concept toute unité lexicale extraite depuis le titre ou le résumé d'un document (publication scientifique), nous appelons par tags toute unité lexicale librement attribuée par les utilisateurs du site de partage de publications scientifiques CiteULike ; et nous appelons par terme toute les unités lexicales sans distinction entre tag ou concept.

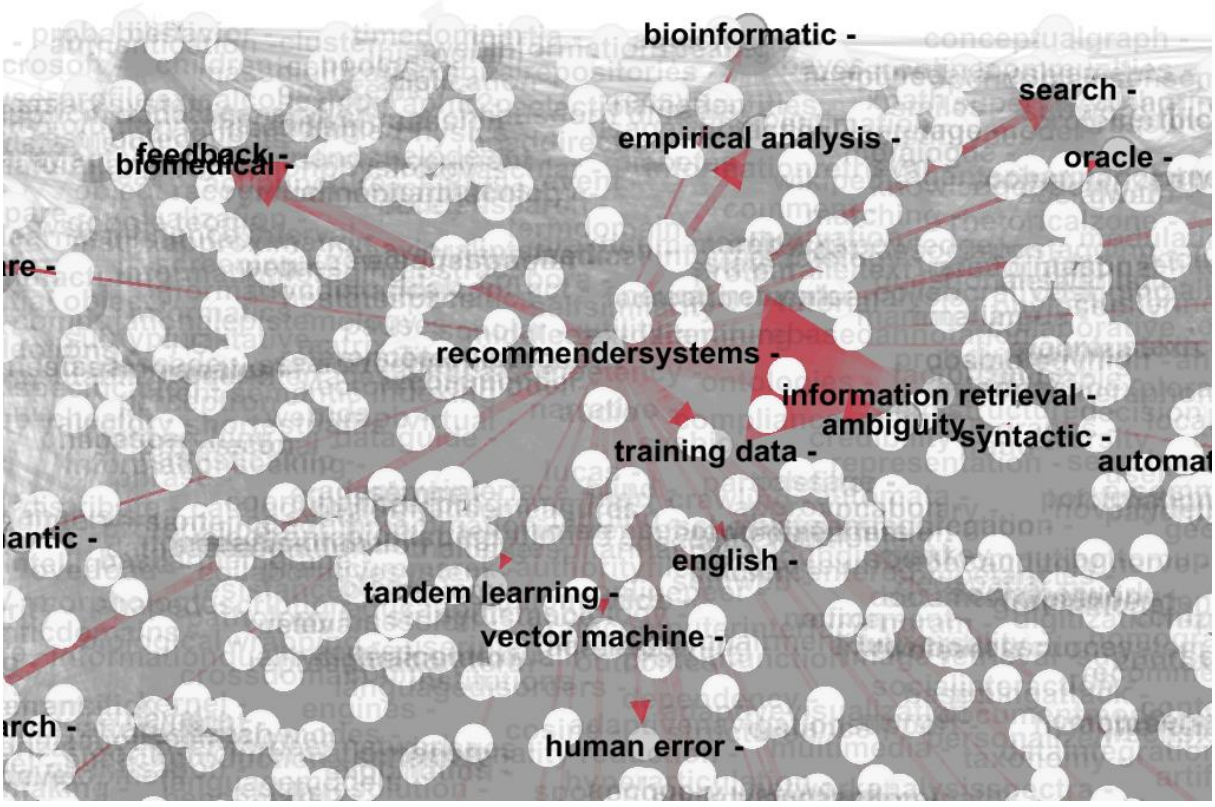


***Figure 2.2*:** une partie de notre graphe (base de données).

3. Etapes de travail

Le travail consiste tout d'abord à collecter un ensemble d'articles à accès restreints représentés chacun par le titre et le résumé. Cette collection constitue notre base de test. Le nombre d'articles est égal à 18 avec la possibilité d'enrichissement de la base de test. Ci-dessous les différents traitements appliqués à cette base de test.

3.1.Prétraitement :

A cette étape nous avons extrait le titre, le résumé et les tags à partir de chaque document dans notre base de test. L'une des principales formes de prétraitement consiste à filtrer les mots inutiles, appelés mots vides de sens.

- Ensuite, nous avons fait une analyse lexicale sur les documents, cette tâche consiste à découper le document en unîtes lexicales ou termes.
- Apres, nous avons supprimé les mots vides (Stopwords) qui n'ajoutent pas beaucoup de sens à un document. A cette fin nous avons utilisé une liste de 1424 mots vides.

nous n'avons conservé que les termes importants dans le document. Cette étape permet de gagner du temps et de réduire la taille de ces documents.

Exemple de mots vides :

- Pronoms :anything, anybody, any, …etc.
- Adjectifs :alone, best, different, … etc.
- Adverbes :downward, each, definitely,…etc.
- Mots répétés dans tous les articles :paper, experiments, methodology,….etc.

- Enfin, la dernière tache de prétraitement, consiste à regrouper les tags morphologiquement similaires et choisir un seul tag représentant et éliminer les autres.

3.2.Calcul des poids :

Ici nous avons calculé pour chaque terme son poids qui mesure son importance dans le document, en utilisons deux formules :

3.2.1. TF IDF (term frequency, inverse document frequency)

Cette formule proposée par [47], [48] prend en compte la fréquence locale d'un terme dans le document (term frequency, TF) et sa fréquence globale dans les autres documents (dans la collection) (inverse document frequency, IDF). Ainsi un mot qui se répète dans tous les documents devient moins important [41]. Le poids d'un terme T dans un document D est obtenu comme suit :

$$TF\ IDF\ =\ TF(T,D)\ \times\ IDF \tag{2.1}$$

Avec :

$$IDF\ = log\ \frac{Le\ nombre\ total\ de\ document}{Le\ nombre\ de\ document\ contenent\ le\ mot} \tag{2.2}$$

Et la fréquence d'apparition d'un mot dans le texte est égale au nombre d'occurrences de ce mot divisé par le nombre total de mots du texte [40].

$$TF = \frac{Nombre\ d'occurrences\ du\ mot\ dans\ le\ texte}{Nombre\ total\ de\ mots\ du\ texte} \tag{2.3}$$

3.2.2. TFP (Term Frequency and Position):

La formule TFP [45] vise à réduire le problème de la longue traîne en favorisant les étiquettes rarement utilisées qui apparaissent dans les sections principales du document. La formule prend en compte le nombre d'occurrences du terme dans le document et son degré d'importance avec une préférence attribuée aux termes apparaissant dans les sections (titre, résumé) [45].

La pondération TFP de chaque étiquette est calculée avec l'équation suivante :

$$TFP = log(1 + \mathrm{TF}) \times \lambda \quad (2.4)$$

Nous avons expliqué TF précédemment et sa formule (2.3).

Dans [45] λ représente un seuil décrivant la position de tag et il prend les valeurs suivantes:

λ = 2 si le tag apparaît dans le titre et le résumé.

λ = 0.5 autrement.

Pour notre cas, et puisque tout les concepts extraits à partir de la base de test sont soit extrait du titre ou du résumé d'un document nous allons ignorer le seuil λ qui est égale à 2 toujours dans notre cas.

Alors nous calculons TFP comme suit :

$$TFP = log(1 + \mathrm{TF}) \quad (2.5)$$

A la fin de l'étape de prétraitement nous avons obtenu pour chaque document une liste de termes ainsi que leur poids (TFIDF /TFP).

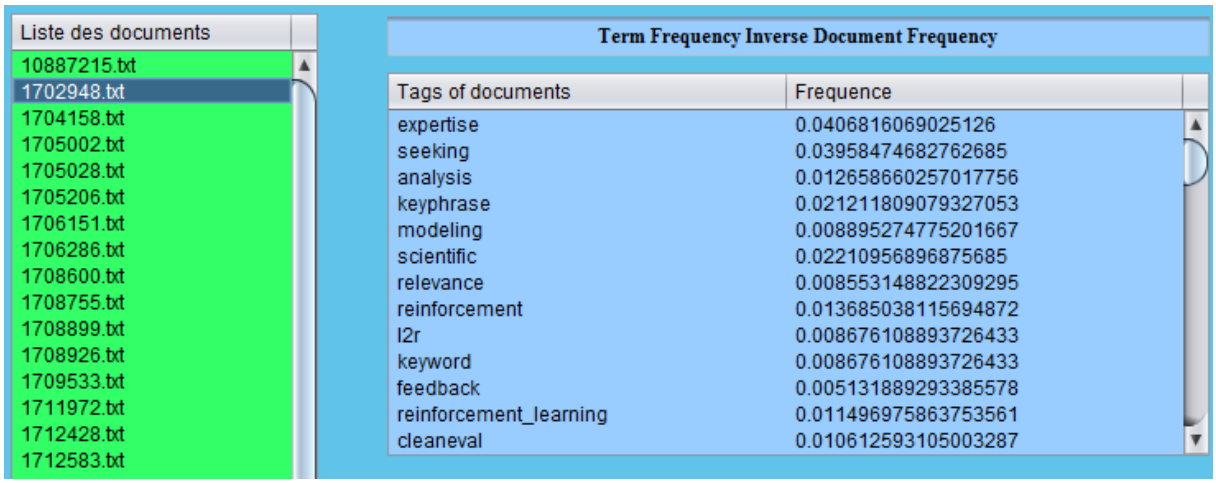


Figure 2.3 **:**la liste des termes avec leurs poids TFIDF après prétraitement

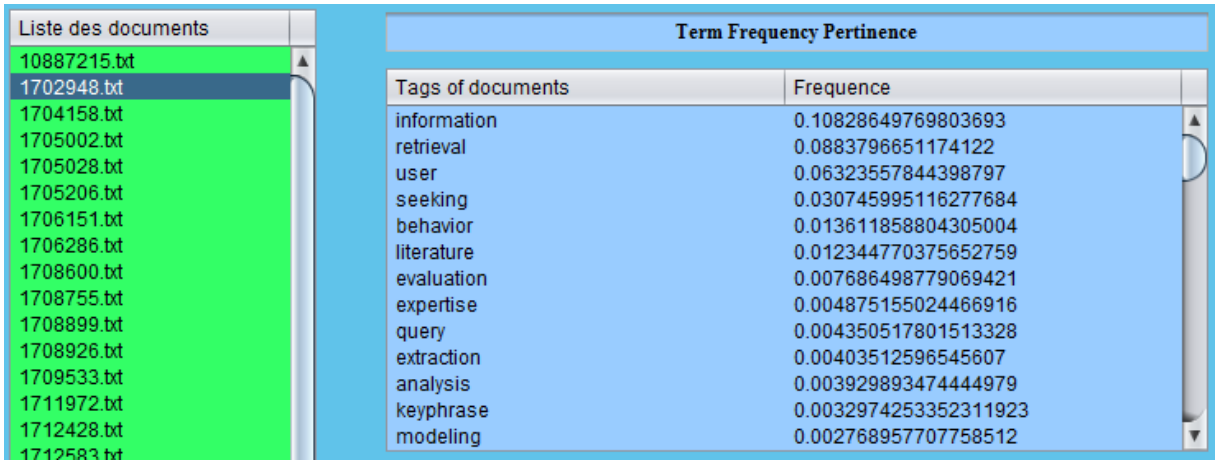


Figure 2.4**:**la liste des termes avec leurs poids TFP après prétraitement

3.3.Algorithme de recommandation:

Notre algorithme de recommandation est basé sur le principe de l'algorithme de Dijkstra (de recherche du plus court chemin) mais nous l'avons adapté selon nos besoins. Les étapes suivies sont les suivantes:

- Tout d'abord, avec le logiciel Gephi nous avons exporté les données du graphe vers un fichier texte. Les données sont organisées sous forme de triplets (Source, target, weight) exemple : (recommendersystems;information retrieval;20.001436548735143). Ensuite, nous avons transformé le fichier en matrice de trois colonnes et 20101 lignes pour faciliter le parcours du graphe.
- Puis, pour chaque document de la base de test nous avons pris les concepts déjà extrait (à l'étape de prétraitement) un par un. Nous avons exploré le graphe en cherchant le nœud contenant le concept. Une fois trouvé le concept du document est sélectionné et

considéré comme nœud initial pour rechercher les chemins possibles qui sortent de ce nœud.

Exemples :

```
les chemins possibles sourtent de concept : recommender :
le chemin 1 : recommender system@explanation@recommendation@system@information
le chemin 2 : recommendation@explanation@information retrieval@system@feedback
le chemin 3 : information retrieval@explanation@feedback@system@hybrid@network
le chemin 4 : feedback@explanation@knowledge management@system@e-commerce@
le chemin 5 : knowledge management@explanation@e-commerce@
le chemin 6 : e-commerce@
```

***Figure 2.5* :** Représente les chemins possibles sortent d'un concept (recommender)

- l'algorithme de Dijkstra cherche à minimiser la valeur finale calculée à partir des poids des arêtes d'un chemin, dans notre cas nous avons cherché le chemin ayant un poids maximal. Ensuite, les chemins sont triés selon leur poids maximaux descendants.
- Pour la sélection des termes à recommander, nous avons extrait pour chaque chemin les trois premiers nœuds. Cette étape a été répétée avec tous les concepts du document à tester.
- Enfin, nous avons obtenu une liste de tags à recommander pour chaque document après avoir éliminé les redondants. Le nombre total de tags recommandés est limité aux 30 premiers tags au maximum et le reste des tags a été filtré.

4. Conception du système:

Le langage de modélisation que nous avons utilisé est Unified Modeling Language (UML) pour présenter la conception de notre application, est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser, spécifier, construire et documenter la conception d'un système.

Nous avons choisi de modéliser notre application par les quatre diagrammes suivants:

- Diagramme de cas d'utilisation.
- Diagramme de classe.
- Diagramme d'activité.
- Diagramme de séquence.

4.1.Diagramme de cas d'utilisation :

Un diagramme de cas d'utilisation est une représentation graphique des interactions entre les éléments d'un système.

Un cas d'utilisation est une méthodologie utilisée dans l'analyse de système pour identifier, clarifier et organiser les exigences du système. Dans ce contexte, le terme "système" désigne quelques choses en cours de développement ou d'exploitation [42].

On a un seul acteur (utilisateur) qui interagit avec le système, les actions faites entre l'utilisateur et le système :

- **Ajout d'un document :** c'est la première association pour ajouter un nouvel article à annoter par la saisie de son titre et son résumé.
- **Evaluer le résultat :** c'est la deuxième association pour récupérer les tags recommandés par le système et donner un jugement de pertinence pour chacun.

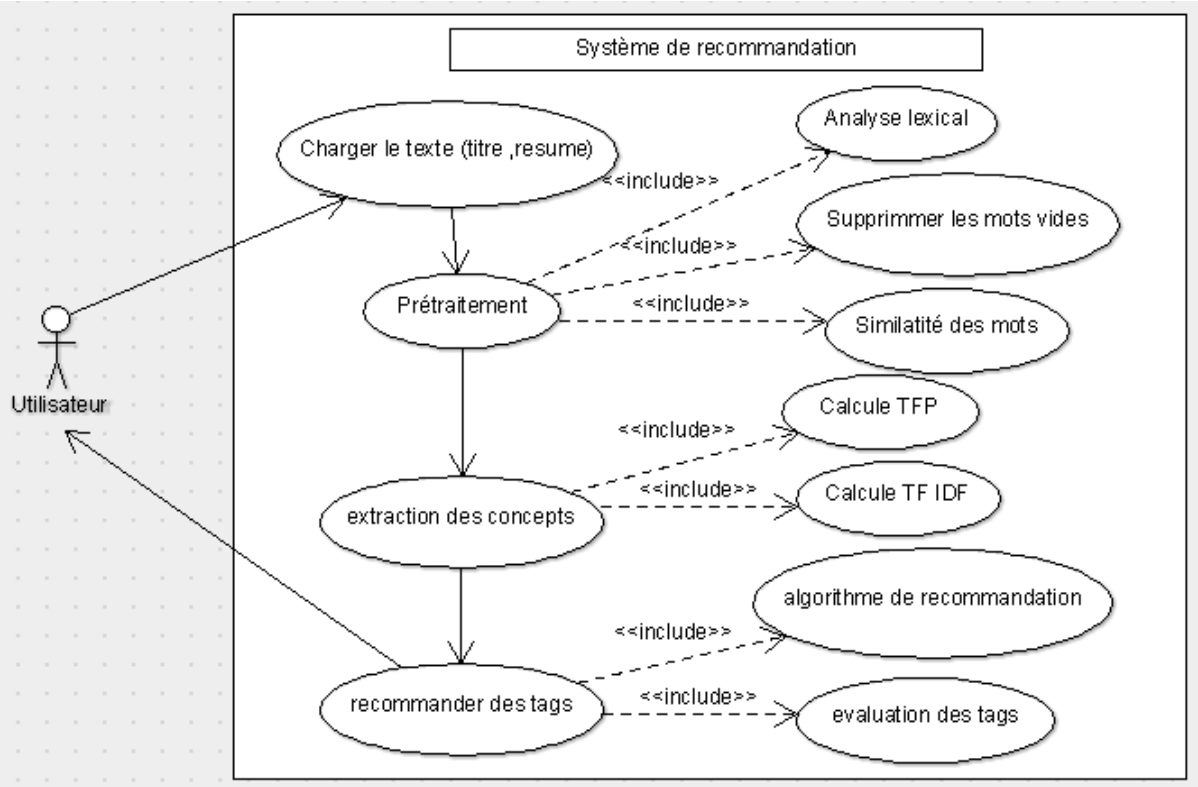


***Figure 2.6*:** diagramme de cas d'utilisation

4.2.Diagramme de classe :

Le diagramme de classes est l'un des types les plus populaires en langage UML. Très utilisé par les ingénieurs logiciels pour documenter l'architecture des logiciels, les diagrammes de classes sont un type de diagramme de structure, car ils décrivent ce qui doit être présent dans le système modélisé [43].

On a quatre classes dans notre diagramme, comme suit :

- **Classe utilisateur :** comporte la méthode **ajouter_texte()** pour ajouter un texte.
- **Classe texte : (titre, résumé)** sont les attributs de cette classe**.**
- **Classe système : (doc, tags_recommender)** sont les attributs de cette classe et la méthode **recommendeur()** permet de récupérer le document, appliqué l'algorithme de sélection et afficher les tags recommandés.
- **Classe recommandation : tags** est un attribut de cette classe

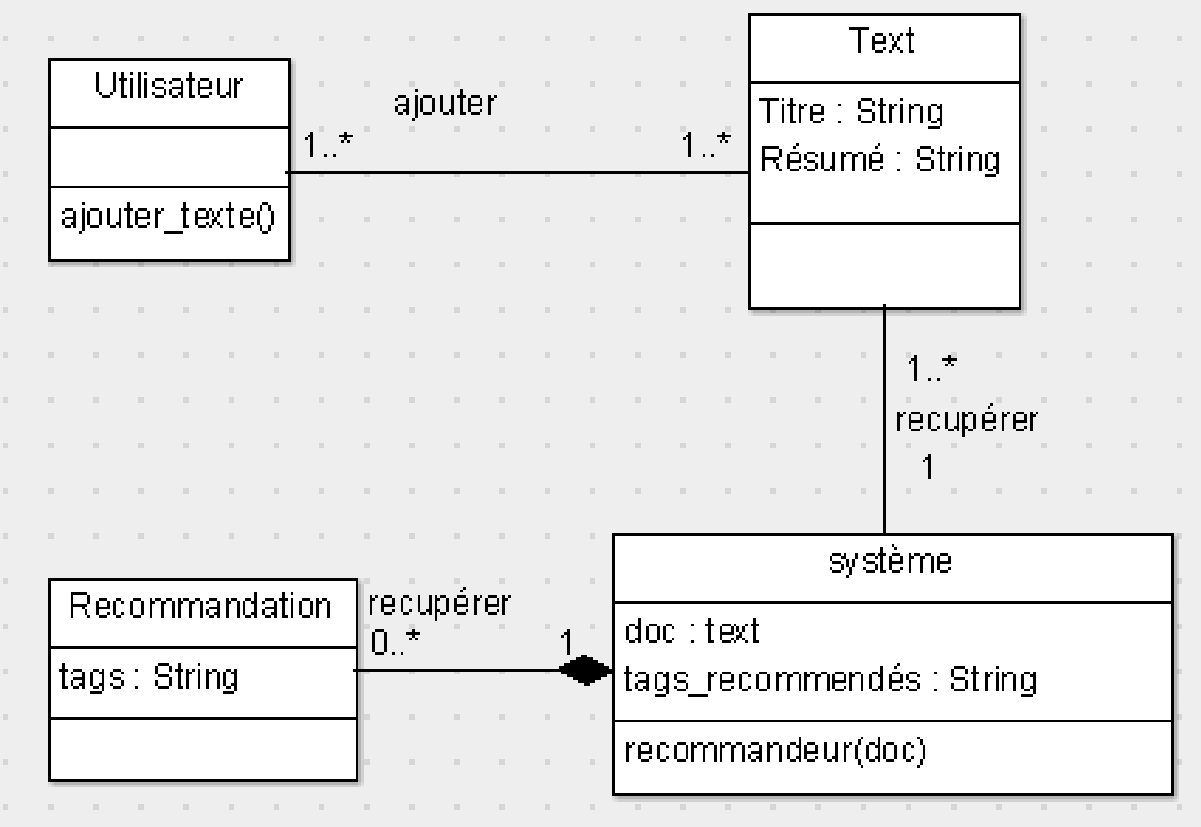


***Figure 2.7*:** Diagramme de classe de notre système

4.3.Diagramme d'activité :

Le diagramme d'activité doit représenter l'ensemble des actions et traitements réalisées par le système, avec tous les branchements conditionnels [43].

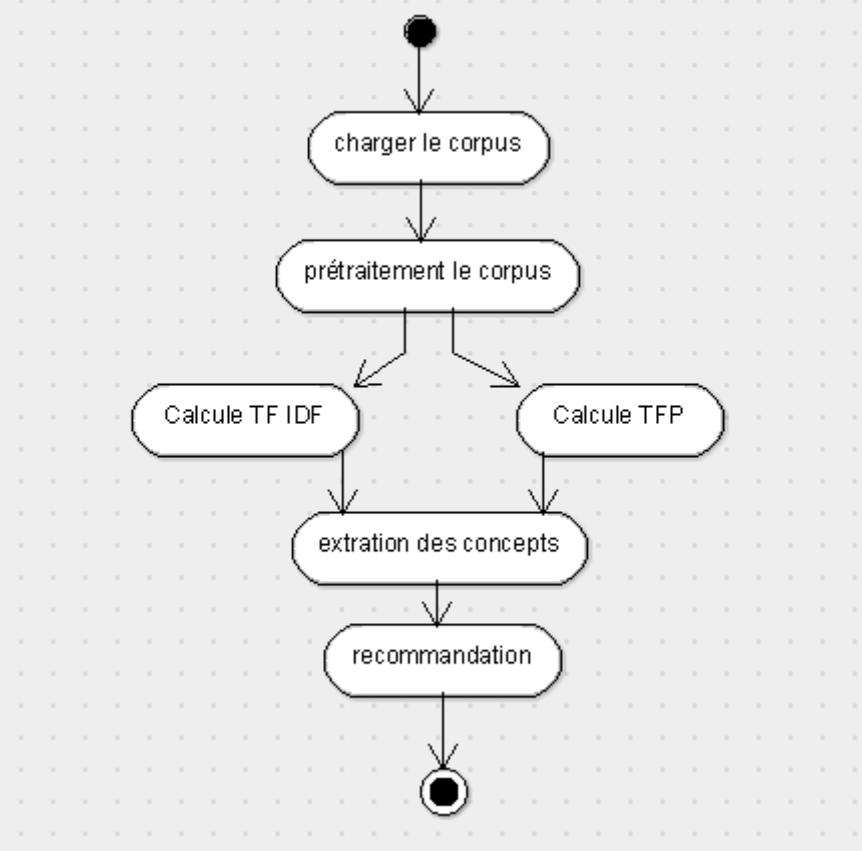


***Figure 2.8*:** Diagramme d'activité de notre système

4.4.Diagramme de séquence :

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique (Message simple, avec durée de vie, message synchrone, message asynchrone, message dérobant).

La figure suivante représente la réalisation d'un diagramme de séquences système qui décrit le scénario nominal du cas d'utilisation **Ajouter un texte**.

Les interactions entre l'utilisateur, le système et les données sont :

- L'utilisateur saisie le titre et le résumé
- Le système fait un prétraitement sur le texte et l'enregistre dans le fichier texte *Données*.

- Le système fait une extraction des concepts du texte et calcule leurs poids, ensuite lance l'algorithme de sélection pour extraire les tags à recommandés.
- Le système affiche les tags recommandés.

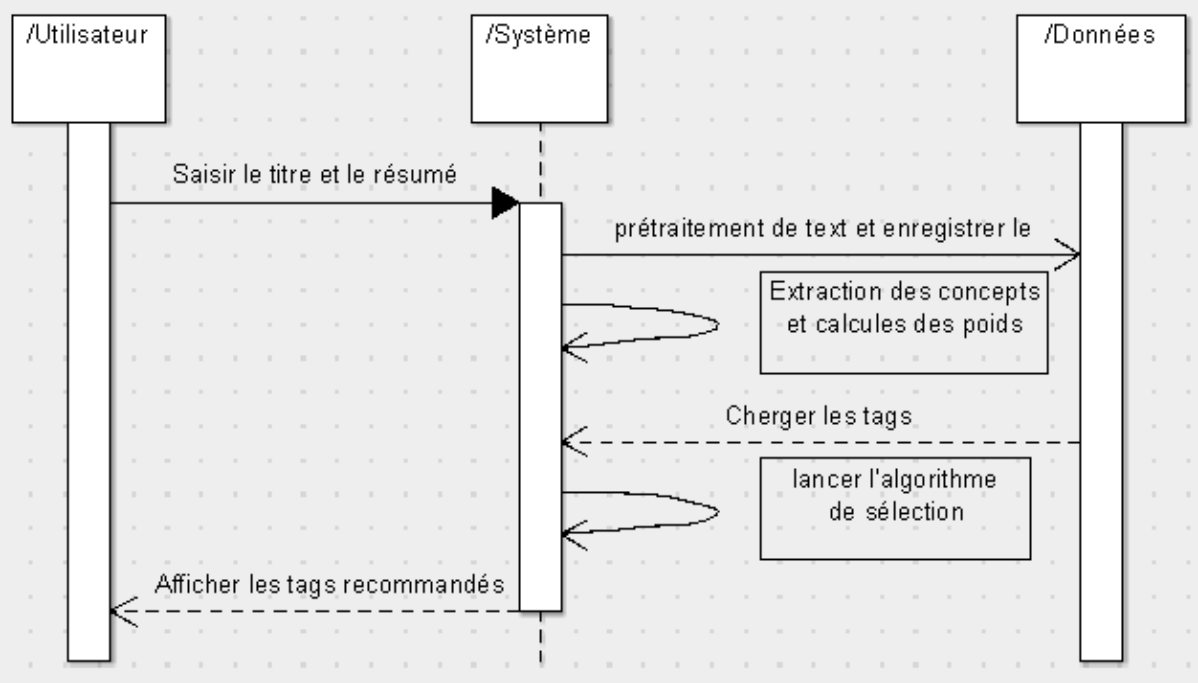


***Figure 2.9*:** Diagramme de séquence

5. Description des classes :

Dans notre application logiciel que nous avons développé en langage java avec plateforme Netbeans, nous avons présenté les classes et les comportements de notre système en deux package qui sont : **Classes et pics.**

- **Package Classes :**

Dans ce package se trouve toutes les classes du système

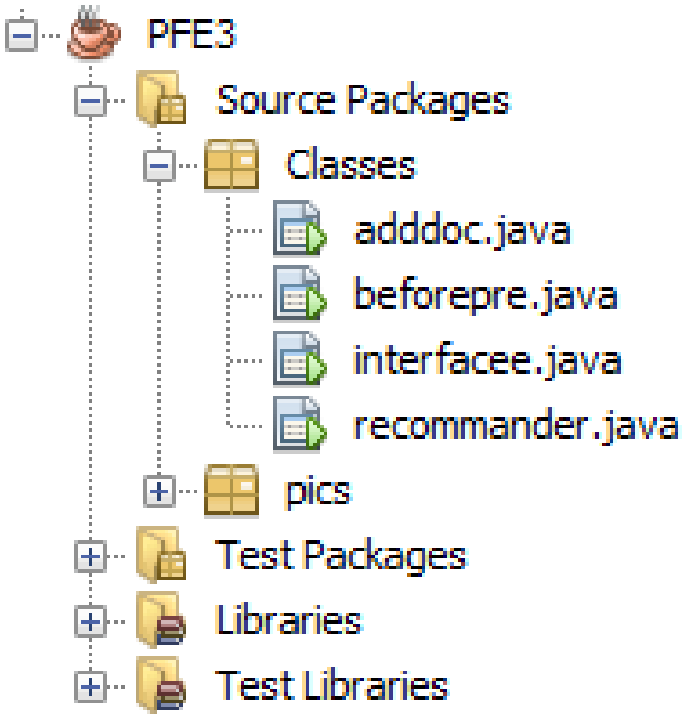


***Figure 2.10*:** Les classes de package Classes de notre application

- La classe **adddoc**, pour l'insertion un nouveau document.
- La classe **beforepre,** affiche les documents avant prétraitement et choisit par quelle formule les poids des concepts seront calculés.
- La classe **interfacee**, affiche l'interface du système
- La classe **remmander**, permet le prétraitement du document, le calcul du TFP et TF IDF des concepts dans chaque document et exécute l'algorithme de sélection.

➢ **Package pics :**

Ce package permet seulement d'afficher les images de design de l'application.

6. Conclusion :

Dans ce chapitre nous avons présenté les différentes étapes méthodologues du travail réalisé. Il a abordé le prétraitement fait sur la base de test, l'algorithme de recommandation et les étapes de la modélisation. Il reste à présenter les détails d'implémentation dans le prochain chapitre.

CHAPITRE 3 : IMPLEMENTATION

1. Introduction :

Dans ce chapitre, nous allons présenter le coté applicatif de notre travail.

Nous citons les outils utilisés tant que l'environnement de développement et le langage de programmation. Nous allons présenter des différentes interfaces de notre application et nous expliquons en détail comment ça fonctionne et nous terminons par les expérimentations et nous discutons les résultats obtenus.

2. Les outils de développement :

Dans cette partie, nous allons spécifier les outils utilisés pour développer notre application :

2.1.Plateforme matérielle :

L'implémentation de l'application est réalisée sur un ordinateur portable ayant les caractéristiques suivantes :

- Machine : HP
- Processeur : Intel(R) Core(TM) i3-4005U
- Fréquence : 1.70GHz
- RAM : 4.00 Go
- Carte graphique : HD Intel(R)
- Système d'exploitation : Microsoft Windows 8.1 Unilingue

2.2.Plateforme logiciel :

2.2.1. Le langage JAVA :

Nous avons choisi JAVA comme langage de programmation pour développer notre application. JAVA est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy. C'est l'un des langages de programmation les plus populaires dans le monde.

Le code Java pouvant être exécuté sur tout système pour lequel la machine virtuelle Java est portée. Cela signifie que les logiciels écrits en JAVA peuvent être exécutés sous Windows, Mac ou Linux sans avoir besoin de les recompiler.

- **JAVA API (application programming interface) :**

  L'interface de programmation d'application (API) Java est une liste de toutes les classes faisant partie du kit de développement Java (JDK). Il inclut tous les packages, classes et interfaces Java, ainsi que leurs méthodes, champs et constructeurs. Ces classes pré-écrites fournissent une quantité énorme de fonctionnalités à un programmeur. Un programmeur doit être au courant de ces classes et savoir comment les utiliser.

2.2.2. Environnement de développement: NetBeans

Nous avons opté pour NetBeans comme un environnement de développement, nous avons utilisé cet environnement de développement intégré(EDI)placé en open source par Sun en juin 2000 sous licence CDDL Common Development and Distribution License) et GPLv2. En plus de Java, NetBeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le PHP et le HTML, ou d'autres (dont Python et Ruby) par l'ajout de greffons. Il offre toutes les facilités d'un IDE moderne (éditeur en couleurs, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Il existe un certain nombre de modules dans NetBeans IDE. GUI (Graphic User Interface) est l'un des meilleurs outils de conception et de développement pour développer des prototypes et concevoir des interfaces graphiques Swing. Cela peut être fait en faisant glisser et en positionnant les composants de l'interface graphique, c'est ce qui le rend trop facile et très rapide.

- La version que nous avons utilisée est NetBeans IDE 8.2.

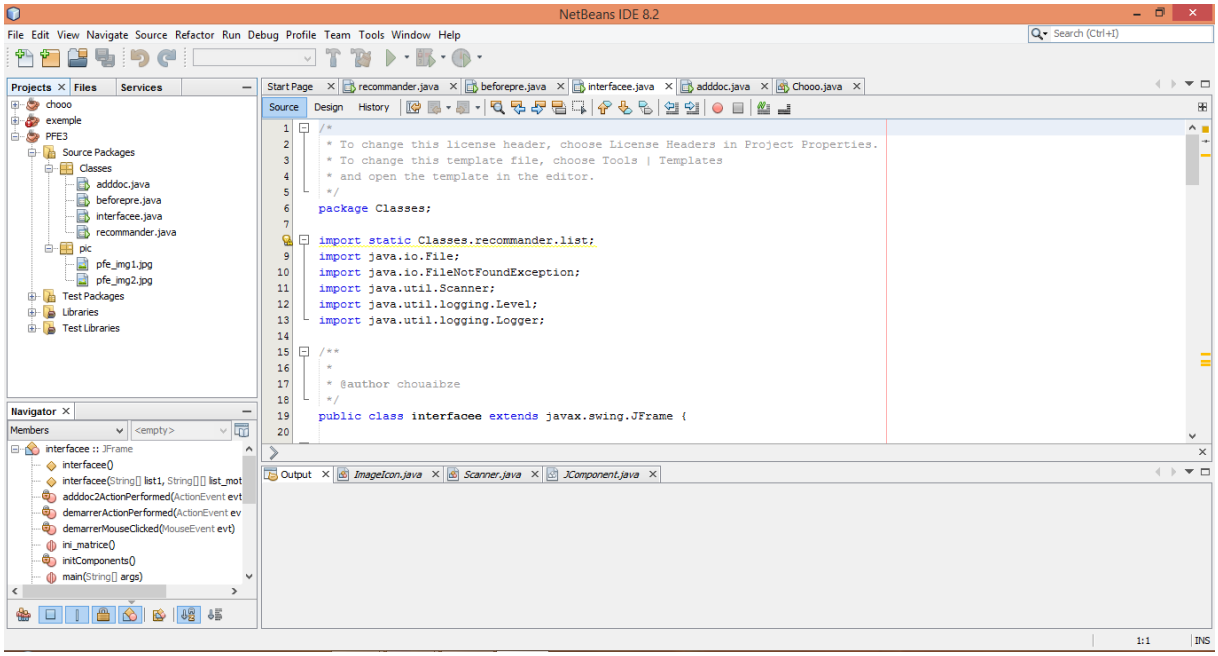


***Figure 3.1* :** Représente l'environnement Netbeans IDE 8.2.

3. CiteULike :

Citeulike qui s'agit d'un site permettant le partage de publications scientifiques a un système de recommandation basé sur les tags. Vous pouvez choisir les tags dont vous avez besoin et en appliquer autant que vous le souhaitez sur un article. En plus, il est possible d'attribuer des tags à plusieurs articles à la fois pour faciliter la recherche d'articles de même contenu, en procédant comme suit :

- 1- Dans la page bibliothèque, cliquez sur la case à cocher et sélectionnez tous les articles que vous voulez classer avec le même tag.
- 2- Ajoutez votre tag.
- 3- CliquezAdd (ajouter).

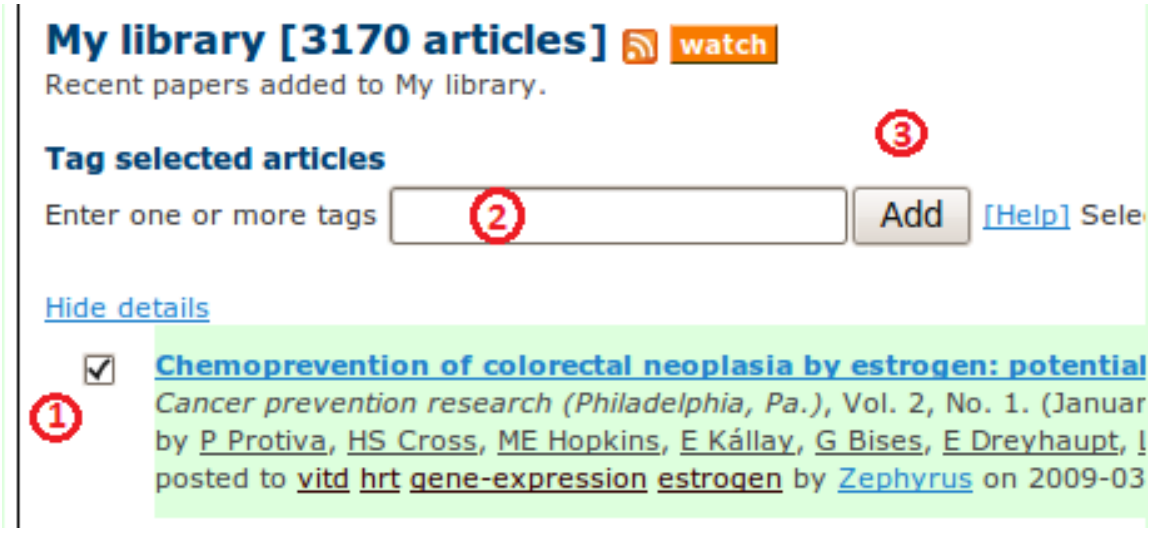


Figure 3.2 : Comment ajouter un tag commun aux articles du même contenu dans le site Citeulike.

Et pour la recherche, vous utilisez le lien 'search' disponible dans la barre d'outils de Citeulike lui-même, vous pouvez rechercher avec un tag ou plusieurs, écrivez votre tags et le système doit recommander des articles ou documents similaires à base de votre tags.

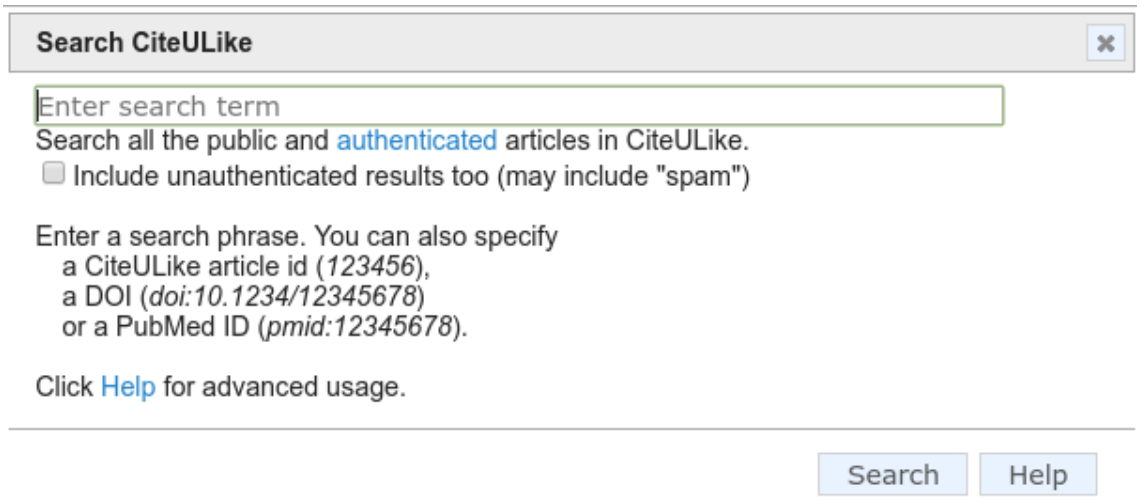


Figure 3.2 : Outils de recherche sur le site Citeulike

4. Description des fichiers utilisés :

Notre base de test contient 18documentshtml en langue anglaise, téléchargés à partir du site web de partage de publications scientifiques « Citeulike ».Ces documents ont été choisi, filtrés et validés par l'expert de domaine qui a conservé seulement les documents du domaine "information retrieval".

Les documents s'agissent de publications scientifiques à accès restreints (payants) c.à.d. sauf le titre, le résumé et quelques métadonnées comme les noms des auteurs, et des éditeurs, le titre de la conférence ou du journal, …etc.

5. Description du système :

Dans cette section, nous présentons les interfaces graphiques de notre application à travers les captures d'écran.

Notre application est dotée d'une interface graphique GUI facile à utiliser qui permet la mise en œuvre des principaux composants de notre système.

Au lancement de l'application, la fenêtre suivante s'affiche:

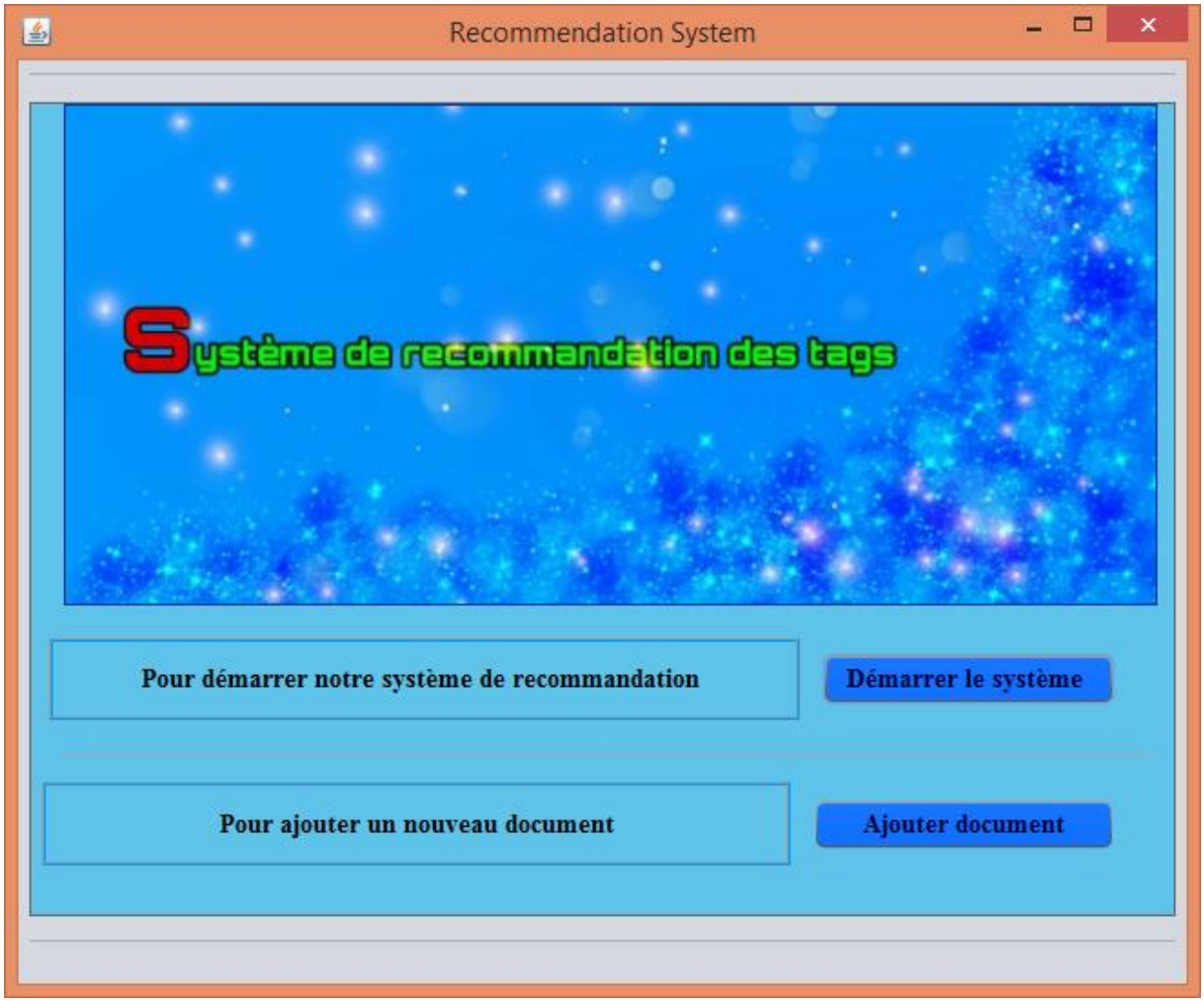


Figure 3.4 : L'interface principale du système

Dans notre interface d'accueil du système, il existe deux boutons qui sont :

- **Démarrer le système :** pour lancer le système de recommandation, et si en cliquant sur ce bouton la fenêtre de la (Figure 3.5) s'affiche.
- **Ajouter document :** ce bouton pour ajouter un nouveau document à notre base de test dont en cliquant sur le bouton la fenêtre de (Figure 3.9) s'affiche.

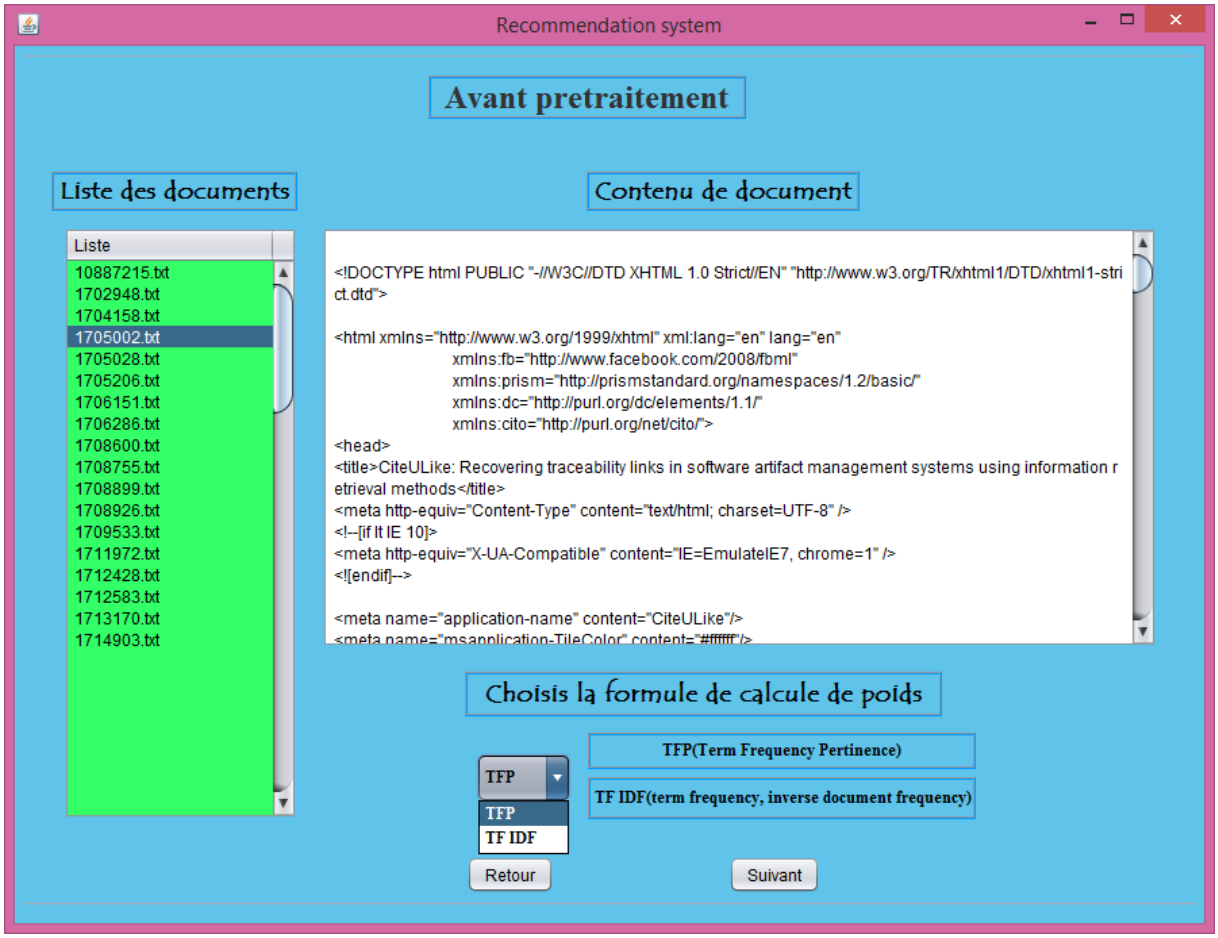


Figure 3.5 : Les documents avant le prétraitement.

Dans cette interface, nous présentons les documents avant le prétraitement, comme nous avons vu dans la Figure3.5 la liste des documents dans le tableau vert et son contenu afficher à coté du tableau, avant de cliquer sur le bouton « suivant » il faut choisir d'abord une formule de pondération pour calculer les poids des termes extraits des documents et il existe deux formules :

- Si on choisit TFP (Term Frequency Pertinence) et on clique sur « suivant » la fenêtre de la (figure 3.6) s'affiche.
- Si on choisit TF IDF (Term Frequency Inverse Document Frequency) et on clique sur « suivant » la fenêtre de la (figure 3.8) s'affiche.

Pour le bouton « retour » permet de retourner à l'interface principale (Figure 3.4).

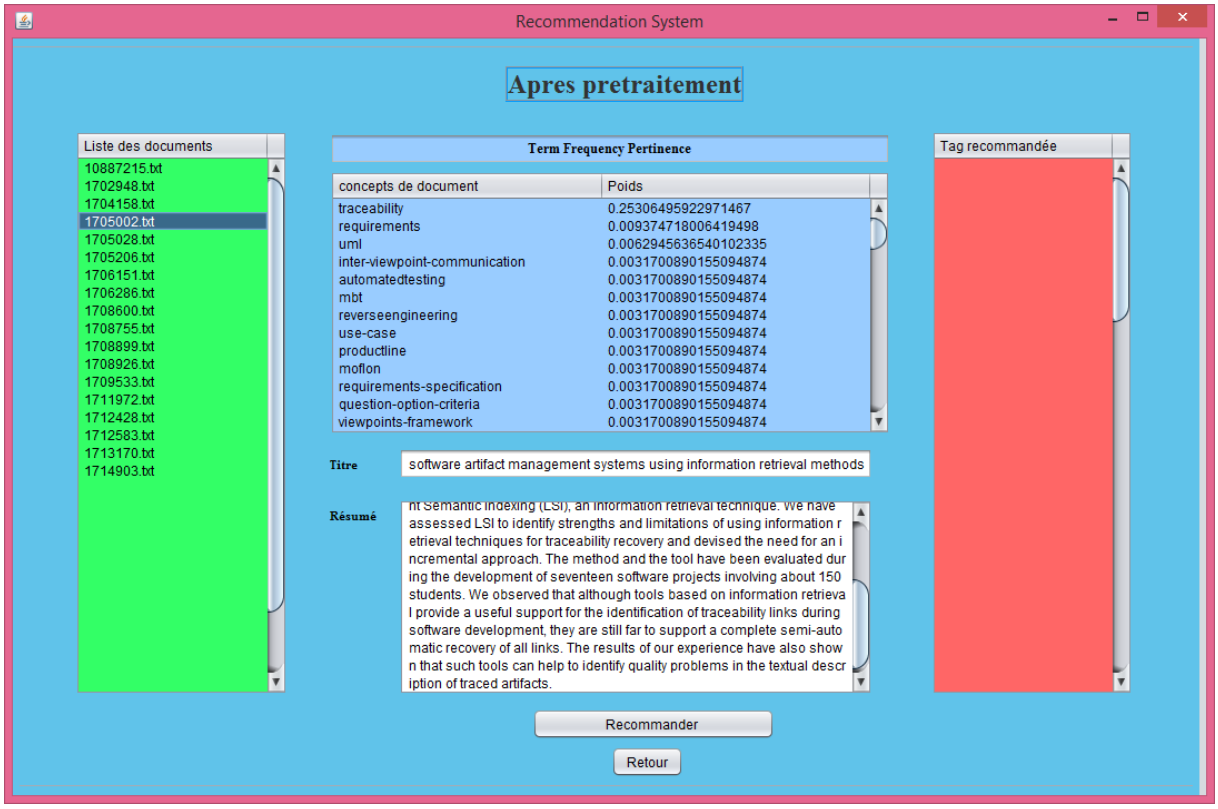


***Figure 3.6* :** Les documents après le prétraitement et calcule des poids avec la formule TFP.

L'interface de la figure 3.6 contient la liste des documents après avoir effectué l'étape de prétraitement et calcule de poids avec la formule TFP.

- Pour chaque document on affiche leur liste de termes, le titre et le résumé.
- Le tableau coloré en bleu, affiche la liste des concepts clés extraits de chaque document. Dès qu'on choisit un document à partir du tableau à gauche le poids de chaque concept qui est calculé avec la formule précédente s'affiche au centre de la fenêtre.
- Le bouton « recommander » permet de lancer l'algorithme de recommandation (l'algorithme de Djikstra appliqué au graphe de termes préalablement crée et décrit dans la section 2.1 du chapitre 2) pour le document sélectionné, de filtrer les tags non significatifs et d'afficher les tags significatifs à recommander dans le tableau rouge comme le montre la figure 3.7.
- Pour le bouton « retour » permet de retourner à l'interface principale (Figure 3.5).

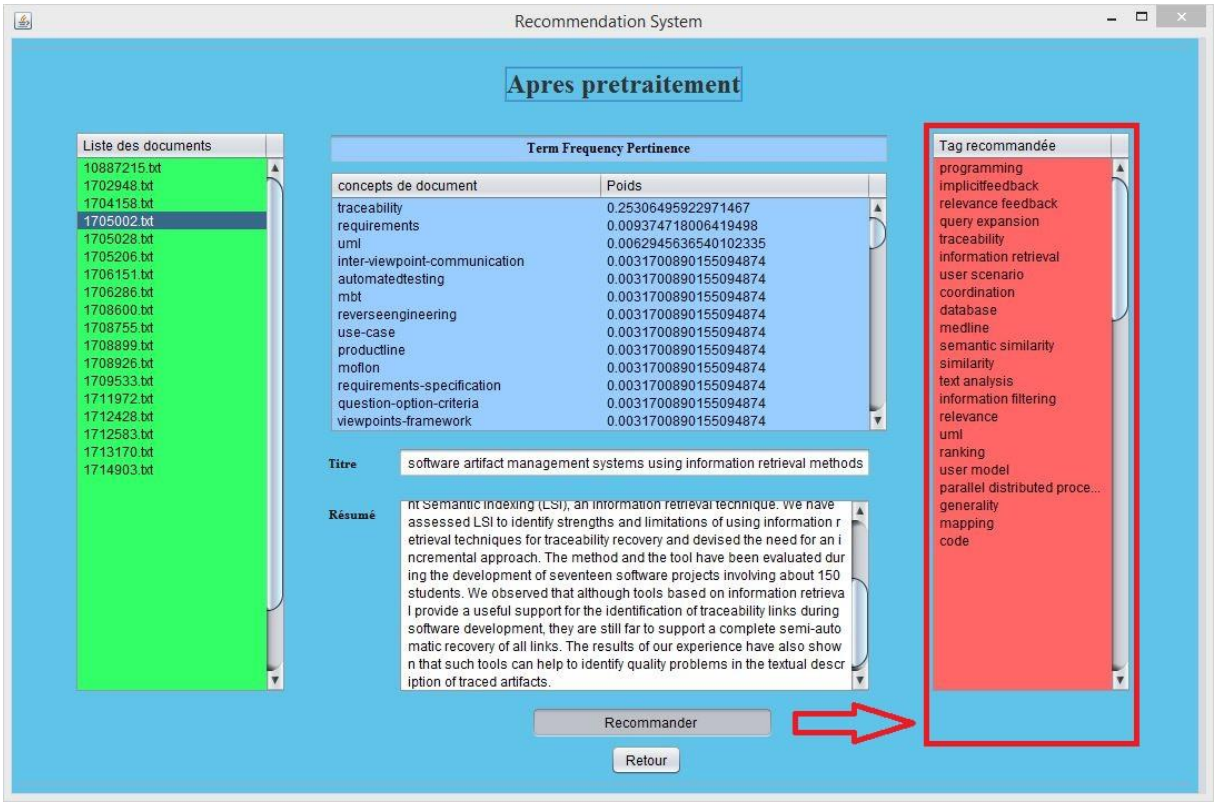


Figure 3.7 : La liste des tags recommandés après avoir cliqué sur le bouton « recommander »

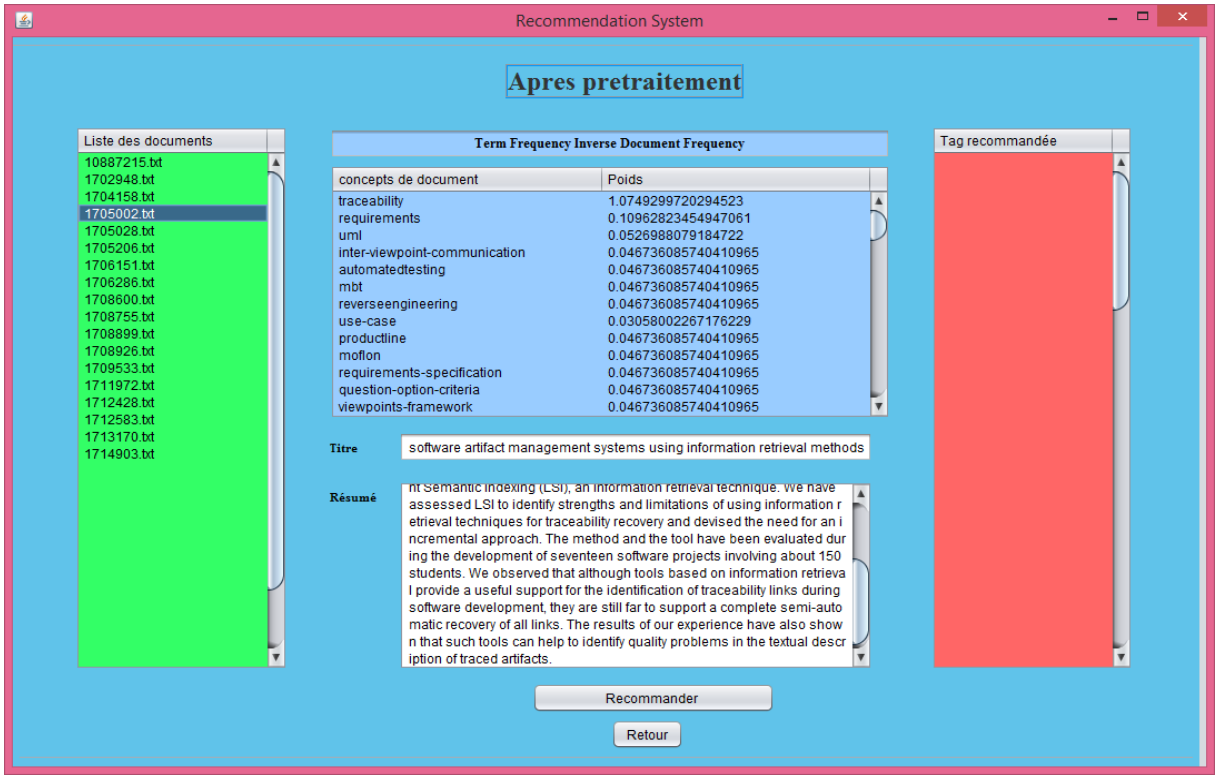


Figure 3.8 : Les documents après le prétraitement et calcule des poids avec la formule TF IDF

Pour l'interface de la figure 3.8affiche les tags recommandés apres extraction des concepts en utilisant la formule de pondération TF IDF (la colonne poids dans le tableau coloré bleu).

Pour le bouton « retour » il permet de retourner à l'interface présentée dans la figure 3.3.

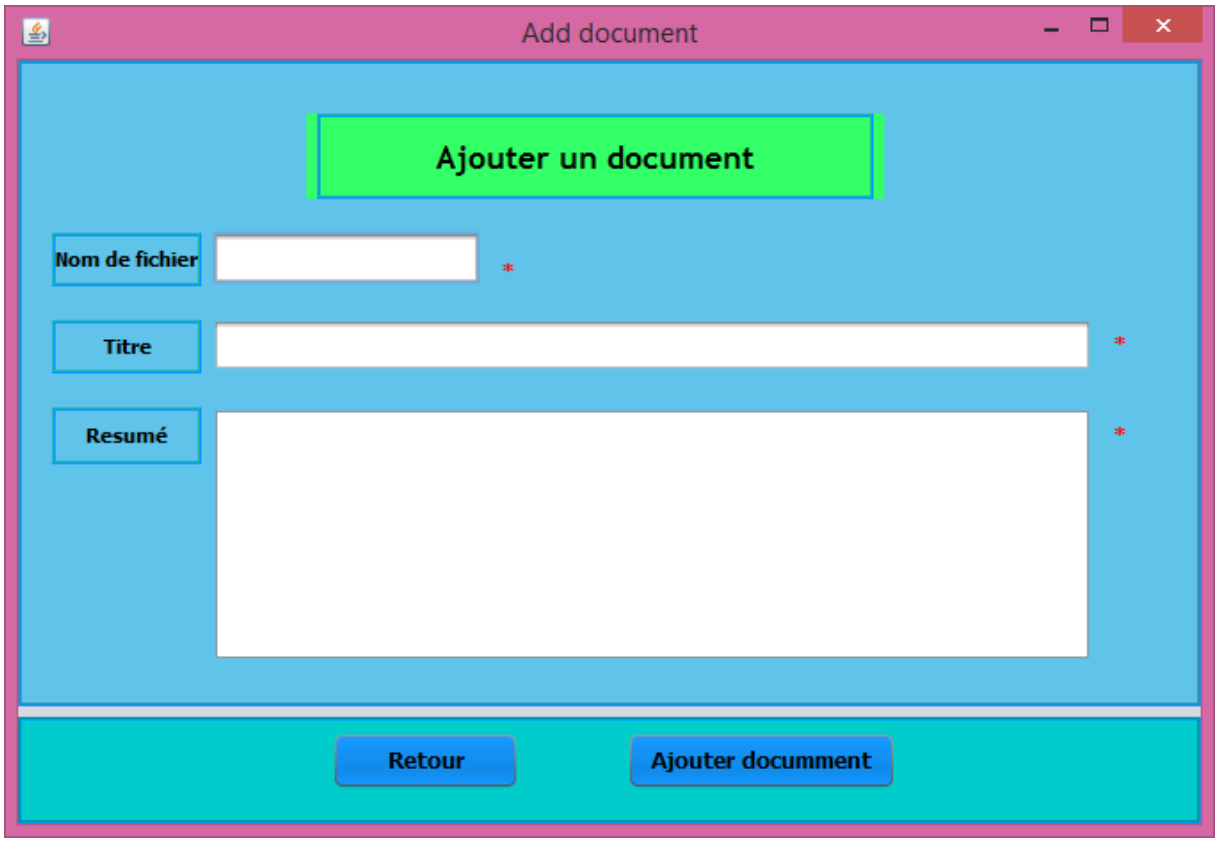


Figure 3.9 : Interface pour ajouter un nouveau document.

Cette interface permet d'ajouter un nouveau document à la base de test.

- Il faut remplir tous les trois champs sinon le document ne sera pas ajouté, l'application informe l'utilisateur pour les remplir tous et la figure 3.11 représente ce cas d'erreur.
- Quand l'utilisateur remplie tout les champs et il clique sur « Ajouter document » l'application l'informe que le document est ajouté avec succès et la figure 3.10 représente ce cas.
- Pour le bouton « retour » il permet de retourner vers l'interface principale du système présentée par la figure 3.4.

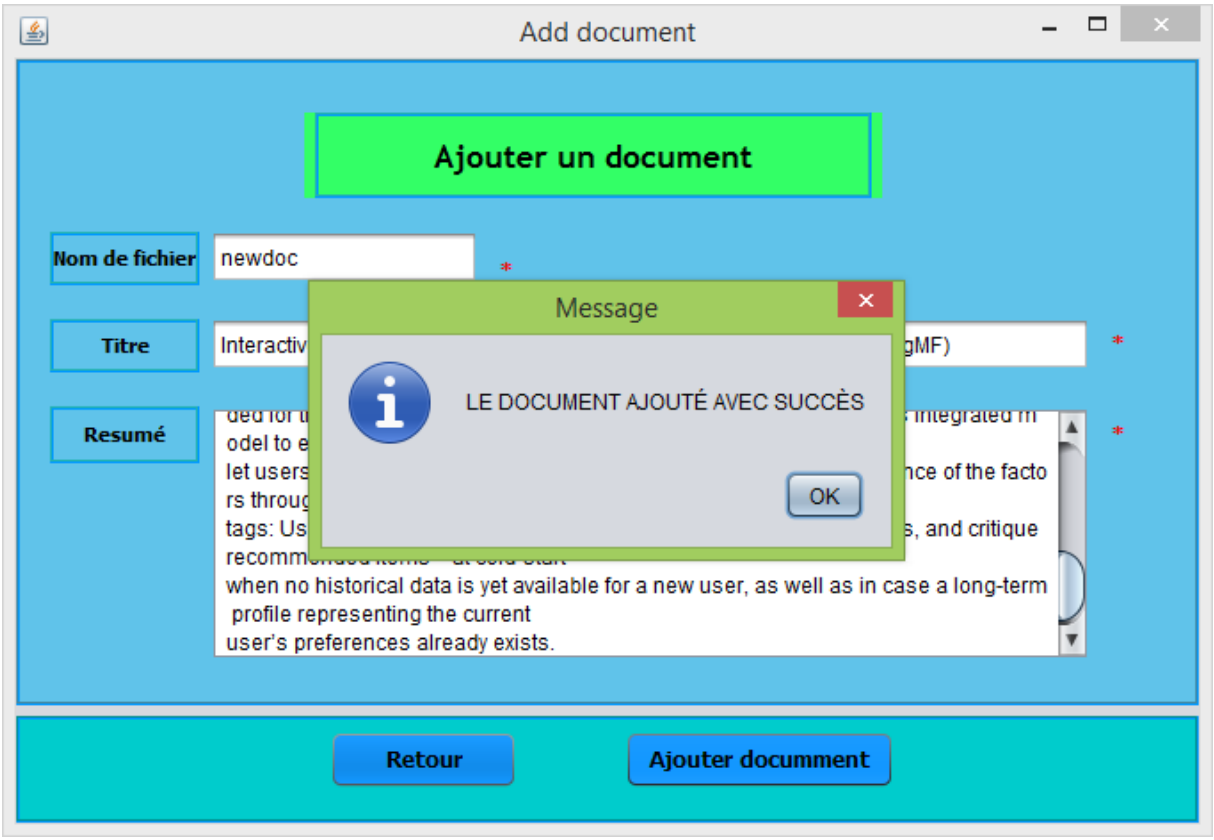


Figure 3.10 **:** L'ajout d'un document avec succès.

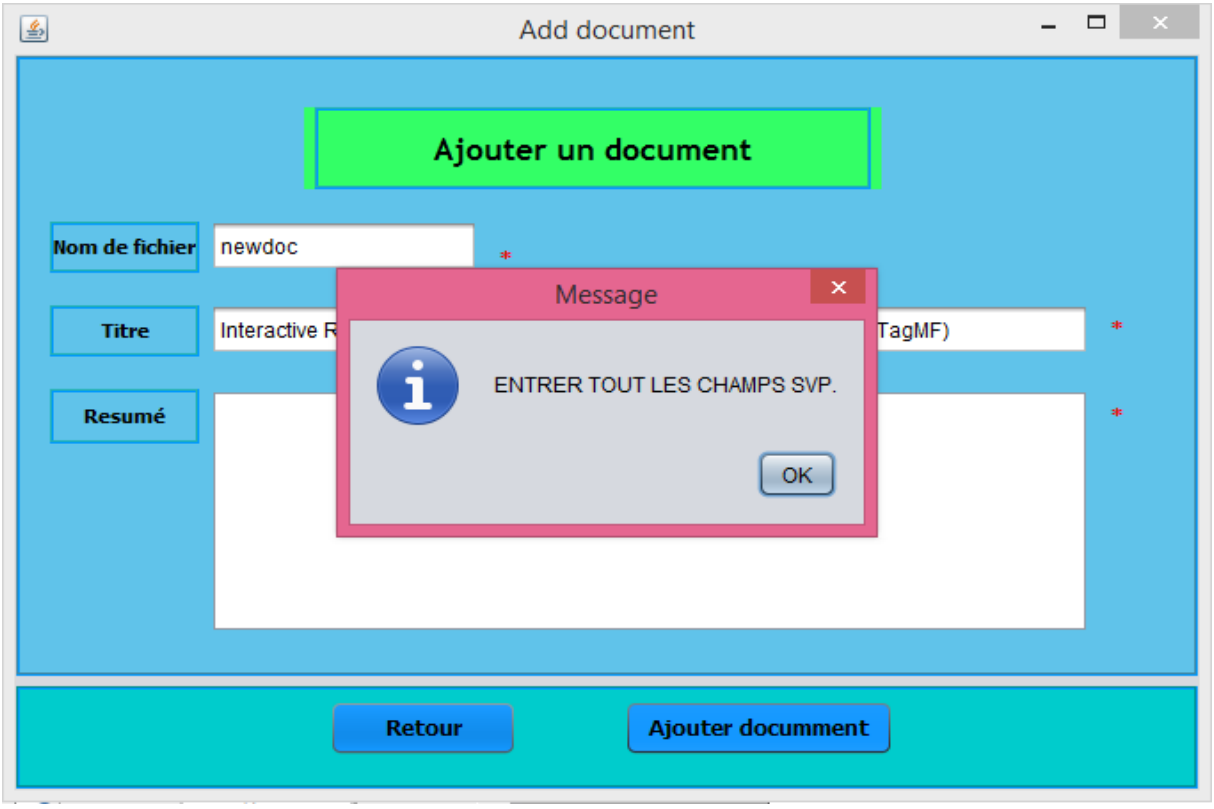


Figure 3.11 **:** Le document n'est pas ajouter car le résumé n'existe pas.

6. Expérimentation et Résultats :

Dans cette partie nous représentons l'évaluation expérimentale de notre application, pour illustrer l'efficacité de notre algorithme de filtrage et de sélection des tags.

- **Evaluation :**

Dans cette section, nous allons évaluer la pertinence des tags recommandés par notre algorithme pour chaque document dans la base de test.

La formule utilisée pour l'évaluation est la « Précision » appliquée sur une collection de documents d'évaluation contenant 18 documents. Il s'agit de faire un traitement qui consiste à rechercher l'existence des tags dans le texte complet de chaque document, donc les tags qui apparaissent dans le texte sont considérés pertinents. Pour ce faire nous avons téléchargé le texte intégral de chaque document dans la base de test pour évaluer si les tags recommandés par le système sur la base du titre et du résumé apparaissent sur le texte intégrale du document scientifique qui est restreint et payant.

$$\text{Précision} = \frac{\text{Nombre de tags pertinents recommandés}}{\text{Nombre total de tags recommandés pour le document}} \quad (3.1)$$

Les résultats obtenus sont résumés dans les trois tableaux ci-dessous :

- Le nombre total de tags recommandés dans ce tableau 3.1, est limité aux 30 premiers tags au maximum et le reste des tags a été filtré.

Les documents	Précision@30	
	TFP	TF IDF
Doc 1	0.8	0.79
Doc 2	0.76	0.73
Doc 3	0.86	0.81
Doc 4	0.76	0.75
Doc 5	0.65	0.65

Doc 6	0.75	0.85
Doc 7	0.83	0.86
Doc 8	0.66	0.83
Doc 9	0.7	0.8
Doc 10	0.8	0.8
Doc 11	0.65	0.7
Doc 12	0.63	0.7
Doc 13	0.6	0.73
Doc 14	0.75	0.71
Doc 15	0.66	0.75
Doc 16	0.74	0.76
Doc 17	0.56	0.66
Doc 18	0.76	0.76
Précision moyenne	0.71	0.75

Tableau 3.1 : Evaluation de la précision@30

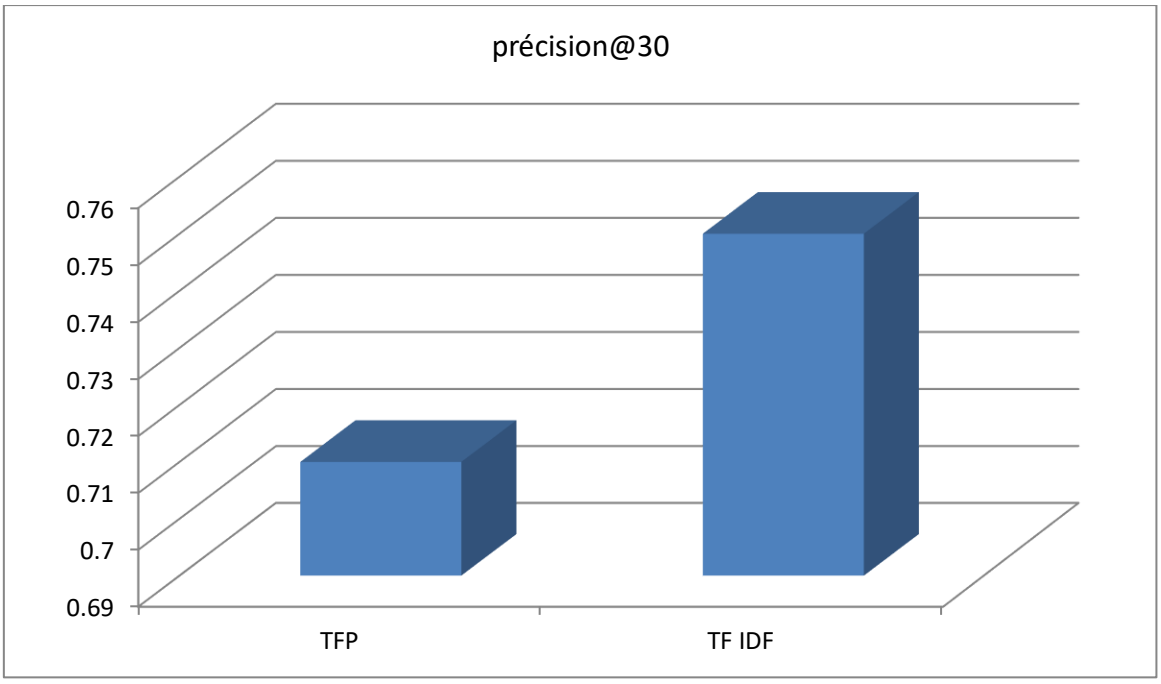


***Figure 3.12*:**La précision moyenne au 30 premiers tags.

- La précision dans ce tableau 3.2, est calculée sur les 20 premiers tags seulement.

Les documents	Précision@20	
	TFP	TF IDF
Doc 1	0.75	0.75
Doc 2	0.7	0.8
Doc 3	0.8	0.85
Doc 4	0.75	0.8
Doc 5	0.65	0.7
Doc 6	0.7	0.85
Doc 7	0.75	0.8
Doc 8	0.8	0.8
Doc 9	0.7	0.9
Doc 10	0.75	0.75
Doc 11	0.65	0.7
Doc 12	0.7	0.7
Doc 13	0.8	0.75
Doc 14	0.85	0.8
Doc 15	0.8	0.81
Doc 16	0.75	0.76
Doc 17	0.7	0.75
Doc 18	0.75	0.7
Précision moyenne	0.74	0.77

***Tableau 3.2* :** Evaluation de la précision@20

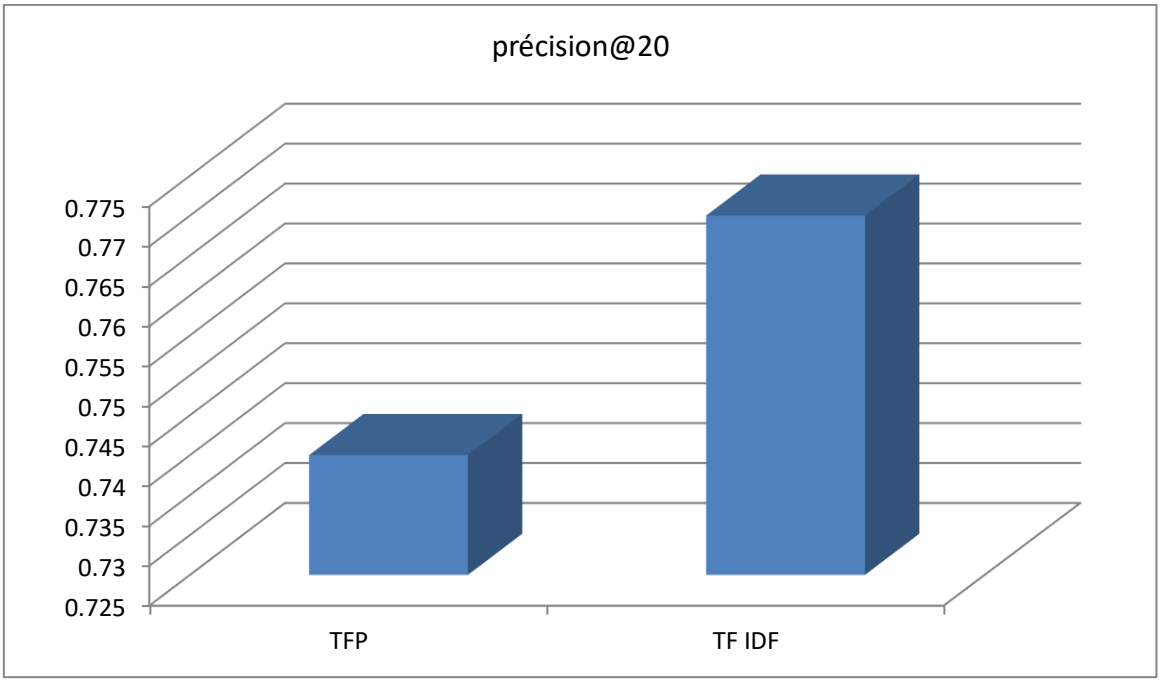


***Figure 3.13* :** La précision moyenne à 20 premiers tags.

- La précision dans ce tableau 3.3, est calculée sur les 10 premiers tags seulement.

Les documents	Précision@10	
	TFP	TF IDF
Doc 1	0.7	0.7
Doc 2	0.7	0.9
Doc 3	0.8	0.8
Doc 4	0.7	0.7
Doc 5	0.6	0.7
Doc 6	0.7	0.8
Doc 7	0.8	0.9
Doc 8	0.8	0.8
Doc 9	0.8	0.9
Doc 10	0.8	0.9
Doc 11	0.7	0.7
Doc 12	0.6	0.7

Doc 13	0.8	0.8
Doc 14	0.8	0.8
Doc 15	0.7	0.7
Doc 16	0.7	0.8
Doc 17	0.8	0.8
Doc 18	0.7	0.7
Précision moyenne	0.73	0.78

***Tableau 3.3* :** Evaluation de la précision@10

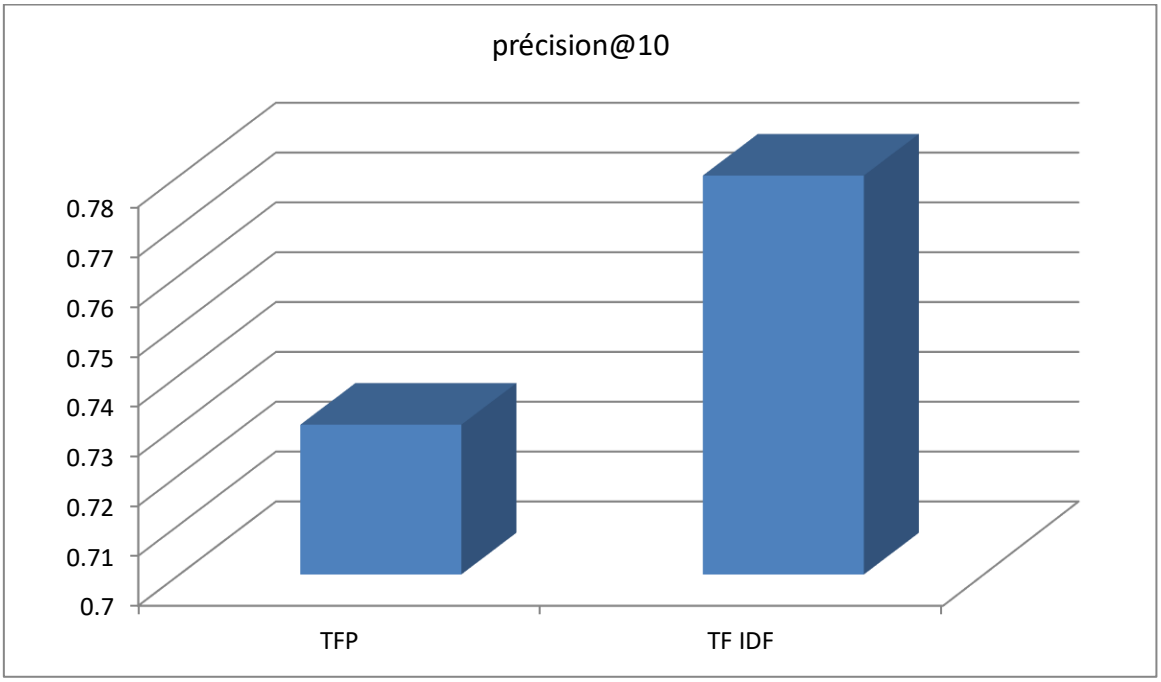


Figure 3.14 : La précision moyenne à 10 premiers tags

- **Discussion des résultats :**

Pour la précision@30: Dans la majorité des cas on trouve que les résultats obtenus sont satisfaisants. Sa valeur minimale était 0.56et maximale était 0.86 en utilisant TFP. Tandis que la valeur minimale en utilisant TFIDF était 0.65 et maximale était 0.86.

Pour la précision @30 le calcul des poids par TFIDF a donné des résultats meilleurs que ceux obtenus par la formule TFP.

Pareil pour la précision@20 qui est égale0.74 en moyenne en utilisant la TFP et à 0.77 en moyenne en utilisant TF IDF, la précision moyenne à 20 a donné des résultats meilleurs que ceux de la précision moyenne à 30. Par exemple, les valeurs de la précision à 20 des tags recommandés pour le doc 14 et le doc 15 étaient plus élevées par rapport aux valeurs de la précision @30 des mêmes documents.

Pour la précision@10:nous trouvons que les valeurs de précision de TF IDF sont supérieures ou égales aux valeurs de précision de TFP pour tous les documents de la base de test.

Les résultats de la précision à 10 obtenus par TF IDF sont également supérieurs pour la précision moyenne en utilisant la TFP (0.78 vs 0.73). Par exemple, pour les doc 7, doc 8 et doc 9 elles étaient très élevées par rapport aux autres valeurs présentées dans les tableaux (3.1 et 3.2).

D'autres part, dans certain cas la TFP donne des résultats supérieurs que ceux donnés par TF IDF car des fois les totaux tags recommandés ne dépasse pas ou égale à 30 dans tableau 3.1 ou 20 dans le tableau 3.2 et ça augmente la précision.

En effet les deux formules ont été appliquées sur les documents de la base de test afin d'attribuer une valeur d'importance à chacun des concepts extrait du document. Cette valeur permet de déterminer par lors de la phase de filtrage et sélection l'ordre des tags recommandés.

En comparant TFIDF et TFP nous constatons que TFP est basé sur la fréquence du terme essentiellement tandis que la TFIDF est basée sur la fréquence de terme dans le document (fréquence locale) et sa fréquence dans toute la collection de documents (fréquence globale). Alors, pour notre problème de filtrage et de recommandation, la prise en compte de la fréquence locale et globale d'un terme est plus pertinent que la prise en compte de la fréquence locale seulement.

Finalement, l'adaptation de l'algorithme de Dijkstra au présent problème de recommandation et de filtrage de tags à base d'un graphe permet d'obtenir des tags significatifs pouvons aider l'utilisateur à comprendre davantage le contenu des publications scientifiques payantes, où le peu de sections en accès libre (titre et résumé) ne permettent pas à l'utilisateur d'évaluer si le contenu intégral de la publication est intéressant ou non.

L'algorithme a réussi à proposé des tags qui existent dans le texte intégral du document payant et à filtrer les tags non significatifs à partir de chaque chemin sélectionné à partir du graphe de termes

7. Conclusion :

Dans ce chapitre, nous avons présenté les outils de développement et les étapes d'implémentation et les différentes interfaces. Nous avons également présenté les résultats des évaluations exprimés en matière de précision @30, @20 et @10. Finalement, nous avons discuté les résultats obtenus. Le but principal de cette implémentation est d'évaluer notre algorithme de sélection et de filtrage qui s'agit d'une adaptation de l'algorithme de Dijkstrade recherche du plus court chemin dans un graphe de termes préalablement crée et développé.

Conclusion générale

Dans le web collaboratif l'aspect de production, de partage et de tagging de ressources a pris une grande ampleur de liberté ce qui a conduit à l'apparition de différentes morphologies d'écriture de mots sur le web. Ces mots ou tags sont utilisés pour décrire le contenu des ressources partagées. Les systèmes de recommandation sont très utiles pour recommander des tags pertinents aux utilisateurs. Cela peut être très utile pour trouver des recommandations nouvelles et fortuites de produits, d'articles ou de tags susceptibles d'intéresser l'utilisateur. L'efficacité du système de recommandation repose sur l'algorithme utilisé pour trouver des ressources intéressantes. Les suggestions de livres sur Amazon ou de films sur Netflix sont des exemples concrets du fonctionnement de systèmes de recommandation. La conception de ces moteurs de recommandation dépend du domaine et des caractéristiques particulières des données disponibles.

Dans ce mémoire nous avons présenté un système de recommandation des tags, écrit en langage de programmation JAVA dans l'environnement IDE netbeans. Le système a été réalisé en plusieurs étapes. Initialement, ce système permet à l'utilisateur de saisir un document représenté par un titre et un résumé pour le rajouter à la base de test, puis on fait un prétraitement pour chaque document dans la base de test et cette opération consiste à extraire les concepts automatiquement de ces documents. À la fin de l'étape de prétraitement, nous obtenons une liste des concepts pour chaque document. Ensuite, nous avons calculé la pondération de chaque concept de document qui mesure l'importance de ce dernier au niveau du document. Nous avons utilisé deux formules pour calculer le poids des concepts qui sont la formule TFP et la formule TF IDF. Puis, pour recommander des nouveaux tags pertinents à l'utilisateur, nous avons exécuté notre algorithme de sélection des tags qui est basé sur l'algorithme de Disjkstra de recherche du plus court chemin sur un graphe de termes préalablement crée. Nous avons obtenu une liste de termes tags à recommander pour chaque document. Enfin, nous avons mené des évaluations des tags recommandés et nous avons discuté les résultats obtenus.

Au terme de ce travail, nous pouvons conclure que l'adaptation de l'algorithme de Dijkstra au présent problème de recommandation et de filtrage de tags à base d'un graphe permet d'obtenir des tags significatifs pouvons aider l'utilisateur à comprendre davantage le contenu des publications scientifiques payantes, où le peu de sections en accès libre (titre et résumé)

ne permettent pas à l'utilisateur d'évaluer si le contenu intégral de la publication est intéressant ou non.

Les résultats obtenus sont satisfaisants, et ils nous encouragent à continuer de travailler davantage sur la qualité des tags recommandés, comme perspectives nous envisageons d'apporter quelques améliorations à savoir :

- L'enrichissement de notre base de test pour obtenir de nouveaux concepts et pour tester l'algorithme de recommandation sur divers documents.
- L'emploi d'une formule de tri de tags à recommander selon un ensemble de paramètres comme le voisinage et la fréquence.

Bibliographie:

[1]: Melville, Prem, and Vikas Sindhwani. "Recommender systems." *Encyclopedia of Machine Learning and Data Mining* (2017): 1059

[2]: Adomavicius, Gediminas, and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." IEEE Transactions on Knowledge & Data Engineering 6 (2005): 734-749.

[3]: P. Brusilovsky, A. Kobsa, and W. Nejdl (Eds.): The Adaptive Web, LNCS 4321, 2007, pp. 291 – 324.

[4]: Goldberg, David, et al. "Using collaborative filtering to weave an information tapestry." *Communications of the ACM* 35.12 (1992): 61-71.

[5]: Breese, John S., David Heckerman, and Carl Kadie. "Empirical analysis of predictive algorithms for collaborative filtering." *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., (1998): pp 43-52.

[6]:Ungar, L. H., and Foster, D. P. Clustering Methods for Collaborative Filtering. In *Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence*, (1998).

[7]: Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J.. Analysis of Recommendation Algorithms for E-Commerce.In *Proceedings of the ACM EC'00 Conference*. Minneapolis, MN. (2000), pp. 158-167

[8]: Desrosiers, Christian, and George Karypis. "A comprehensive survey of neighborhood-based recommendation methods." *Recommender systems handbook*. Springer, Boston, MA, (2011): 107-144.

[9]: Herlocker, Jon, Joseph A. Konstan, and John Riedl. "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms." *Information retrieval* 5.4 (2002): 287-310.

[10] : Resnick, Paul, et al. "GroupLens: an open architecture for collaborative filtering of netnews." *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM, (1994) : 175–186.

[11] : Shardanand, Upendra, and Pattie Maes. "Social information filtering: algorithms for automating" word of mouth"." *Chi*. Vol. 95. (1995) :pages 210–217.

[12] : Mladenic, Dunja. "Text-learning and related intelligent agents: a survey." *IEEE intelligent systems and their applications* 14.4 (1999): 44-54.

[13] : Balabanović, Marko, and Yoav Shoham. "Fab: content-based, collaborative recommendation." *Communications of the ACM*40.3 (1997): 66-72.

[14] : Melville, Prem, Raymond J. Mooney, and Ramadass Nagarajan. "Content-boosted collaborative filtering for improved recommendations." *Aaai/iaai* 23 (2002): 187-192.

[15]: MOONEY, Raymond J. et ROY, Loriene. Content-based book recommending using learning for text categorization. In : *Proceedings of the fifth ACM conference on Digital libraries*. ACM, 2000. p. 195-204.

[16]: Pazzani, Michael, and Daniel Billsus. "Learning and revising user profiles: The identification of interesting web sites." *Machine learning* 27.3 (1997): 313-331.

[17]: Burke, R.. Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, 12(4), (2002):331–370.

[18]: Herlocker, Jonathan L., et al. "Evaluating collaborative filtering recommender systems." *ACM Transactions on Information Systems (TOIS)* 22.1 (2004): 5-53.

[19]: Ricci F, Rokach L, Shapira B, Kantor P , Recommender systems handbook. Springer, Berlin, (2011).

[20]: SCHEIN, Andrew I., POPESCUL, Alexandrin, UNGAR, Lyle H., *et al.* Methods and metrics for cold-start recommendations. In : *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2002. p. 253-260.

[21] : MOONEY, Raymond J. et ROY, Loriene. Content-based book recommending using learning for text categorization. In : *Proceedings of the fifth ACM conference on Digital libraries*. ACM, 2000. p. 195-204.

[22]: Melville, Prem, Raymond J. Mooney, and Ramadass Nagarajan. "Content-boosted collaborative filtering for improved recommendations." *Aaai/iaai* 23 (2002): 187-192.

[23]: JIN, Rong et SI, Luo. A bayesian approach toward active learning for collaborative filtering. In : *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. AUAI Press, 2004. p. 278-285.

[24] : ZIEGLER, Cai-Nicolas, MCNEE, Sean M., KONSTAN, Joseph A., *et al.* Improving recommendation lists through topic diversification. In : *Proceedings of the 14th international conference on World Wide Web*. ACM, 2005. p. 22-32.

[25]: SU, Xiaoyuan, KHOSHGOFTAAR, Taghi M., ZHU, Xingquan, *et al.* Imputation-boosted collaborative filtering using machine learning classifiers. In : *Proceedings of the 2008 ACM symposium on Applied computing*. ACM, 2008. p. 949-950.

[26]: Herlocker, Jonathan L., et al. "Evaluating collaborative filtering recommender systems." *ACM Transactions on Information Systems (TOIS)* 22.1 (2004): 5-53.

[27]:Fayrouz Soualah-Alila. CAMLearn: Une Architecture de Système de Recommandation Sémantique Sensible au Contexte. Application au Domaine du M-Learning.. Informatique. Université de Bourgogne, 2015.pp 59

[28]: Hmimida, Manel. *Une nouvelle approche topologique pour la recommandation de tags dans les folksonomies*. Diss. Conservatoire national des arts et metiers-CNAM, 2015. Pp 35

[29]:Mohamed Nidhal Jelassi. Un système personnalisé de recommandation à partir de concepts quadratiques dans les folksonomies. Autre [cs.OH]. Université Blaise Pascal - Clermont-Ferrand II, 2016,pp 32.

[30]: Toine Bogers : Tag-Based Recommendation, Part of the Lecture Notes in Computer Science book series(LNCS, volume 10100), 2018, pp463-446.

[31]:Gaël Sauvanet. Recherche de chemins multi objectifs pour la conception et la réalisation d'une centrale de mobilité destinée aux cyclistes. Modélisation et simulation. Université François Rabelais - Tours, 2011, Pp 40 41

[32]: https://www.techopedia.com/definition/4927/web-collaboration ,06 juin 2019.

[33]: PUTHIYA PARAMBATH, Shameem A., USUNIER, Nicolas, et GRANDVALET, Yves. A coverage-based approach to recommendation diversity on similarity graph. In : *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016. p. 15-22.

[34]: LI, Xin et CHEN, Hsinchun. Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. *Decision Support Systems*, 2013, vol. 54, no 2, p. 880-890.

[35]: Zhang, Yuan, Ning Zhang, and Jie Tang. "A collaborative filtering tag recommendation system based on graph." *ECML PKDD discovery challenge* (2009): 297-306.

[36]: Ma, Hao, Irwin King, and Michael R. Lyu. "Mining web graphs for recommendations." *IEEE Transactions on Knowledge and Data Engineering* 24.6 (2011): 1051-1064.

[37]: MOONEY, Raymond J. et ROY, Loriene. Content-based book recommending using learning for text categorization. In : *Proceedings of the fifth ACM conference on Digital libraries*. ACM, 2000. p. 195-204.

[38]: MUSTO, Cataldo, BASILE, Pierpaolo, DE GEMMIS, Marco, *et al.*Automatic Selection of Linked Open Data Features in Graph-based Recommender Systems. In : *CBRecSys@ RecSys*. 2015. p. 10-13.

[39]: Demovic, Lubos, et al. "Movie recommendation based on graph traversal algorithms." *2013 24th International Workshop on Database and Expert Systems Applications*. IEEE, 2013.

[40]: J. Zidén, « A study of term weighting approaches for short-length documents ». Février 2008.

[41]:S. Saldarriaga, « Approches textuelles pour la catégorisation et la recherche de documents manuscrits en-ligne ». 24 mars 2010

[42]: https://whatis.techtarget.com/definition/use-case-diagram ,15 juin 2019.

[43]: Roques, Pascal, and Franck Vallée. *UML 2 en action: De l'analyse des besoins à la conception J2EE*. Eyrolles, 2004, pp 37.

[44]:Khobizi Abdennour,"Une Approche d'Indexation Sémantique de Documents Textuels sur le Web Social", Juin 2017

[45]:Boughareb, D., Farah, N. &Seridi, H.. "Positioning Tags Within Metadata and Available Papers‟ Sections: Is It Valuable for Scientific Papers Categorization? ". In: the 5th International Conference on Control Engineering and Information Technology, *Proceeding of Engineering and Technology –PET,* Tunisia. Vol 32 (2017) :(pp. 57-61).*

[46] :Waad GASMI, Mémoire Maitrise : «Le Filtrage Basé sur le Contenu pour la Recommandation de cours(FCRC)», Département de génie informatique et génie logiciel, 2011.

[47] : Salton, G. Search and retrieval experiments in real-time information retrieval.IFIP Congress (2) (1968): 1082-1093.

[48] : Jones, K. Sparck. "Experiments in relevance weighting of search terms." *Information Processing & Management* 15.3 (1979): 133-144.

[49] Lien de téléchargement gephi: https://gephi.org/users/download/ .

[50] http://www.ariadne.ac.uk/issue/51/emamy-cameron/ , 20 mars 2019.