Deep learning pour la recherche des experts scientifiques

HOUATI Chakib Mouloud, 171731070912
MEZIANI Serine, 171731070904

Septembre 2022

Résumé

La recherche d'experts scientifiques (*scientific expert finding*) est une tâche récurrente dans le milieu académique. En effet, s'exprime souvent le besoin de trouver des encadreurs, membres de jury pour des projets de fin d'études ou de thèses, évaluateurs de propositions de projets de recherches, référés pour des articles soumis à des revues scientifiques ou des membres de comités de programmes de conférences. Cette recherche est rendue très difficile par l'augmentation du nombre de chercheurs et des travaux scientifiques ainsi que par la grande diversification et spécialisation des domaines de recherches scientifiques.

L'objectif de ce travail sera, de concevoir un système de recherche d'experts par des requêtes exprimées en langage naturel. Ce système sera basé sur les données des articles publiés par les chercheurs (titre, résumé, mots-clés). Le système proposé intégrera les techniques les plus récentes de traitement du langage naturel (NLP) basées sur le deep-learning, en particulier la technique d'apprentissage automatique à base de *transformer BERT* (*SciBERT* et *RoBERTa*), développée par *Google*, et publiée en 2018. Nous présenterons trois contributions principales : une nouvelle approche de l'indexation, l'expansion des requêtes à l'aide de la définition, et une méthode de distribution des scores. En plus de cela, la construction d'un nouveau corpus de test basé sur l'ACM.

"Une plateforme nationale de recherche d'experts académiques a été créée à l'issue de ce projet. Ce portail intègre les méthodes que nous avons proposées, et les valide sur le cas de la production scientifique algérienne.

Mots clés :
Scientific Expert Finding ; Deep Learning ; Word Embeddings ; Bibliographic Databases ; Voting Models ; Query Expansion ; BERT ; SciBERT ; RoBERTa ; FAISS.

Abstract

Scientific expert finding is a recurrent task in the academic world. Indeed, it is often necessary to look for supervisors, jury members for end-of-study or thesis projects, evaluators for research project proposals, referees for articles submitted to scientific journals or members of conference program committees. This research is made very difficult by the increase in the number of researchers and scientific work as well as the great diversification and specialization of the scientific research fields.

The objective of this work will be to design an expert finding system that enhanced through the expansion of queries expressed in natural language. This system will be based on the data of the articles published by the researchers (title, abstract, keywords). The proposed system will integrate the most recent techniques in natural language processing (NLP) based on deep-learning, in particular the transformer-based machine learning technique *BERT* (*SciBERT* and *RoBERTa*), developed by *Google*, and published in 2018. We will present three main contributions : a new indexation, expansion of queries by using the definition, and a score distribution method. In addition to this, the construction of a new test corpus based on ACM.

A national platform for expert research was created at the end of this project. This portal integrates the methods that we proposed, and validates them on the case of the Algerian scientific production.

Key words :
Scientific Expert Finding ; Deep Learning ; Word Embeddings ; Bibliographic Databases ; Voting Models ; Query Expansion ; BERT ; SciBERT ; RoBERTa ; FAISS.

Remerciements

It is with genuine gratitude and warm regard that I dedicate this thesis to my loving parents, ***Abdelaziz*** *and* ***Salima****, who have been always by my side throughout my life and especially in my educational journey. To my beloved siblings,* ***Tahar*** *and little* ***Nonat****.*

I would also like to acknowledge my appreciation and gratefulness to my supervisors ***Mr. BELLAZZOUGUI Djamal*** *and* ***Mr. CHAA Messaoud*** *for their valuable advises and orientations that allowed us to carry out our work successfully.*

I sincerely thank the members of the jury, ***Mr. HAMMAL Youcef*** *and* ***Mr. KHENNAK*** *Ilyes for the honor they give us by accepting to judge this work.*

Finally, a special thanks to my aunt ***Souhila****, my partner* ***Chakib****, and to all the people who have reviewed this thesis.*

Remerciements

J'adresse mes sincères remerciements à mes deux encadreurs, ***Mr. BELLAZZOUGUI Djamal*** *et* ***Mr. CHAA Messaoud*** *pour toutes leurs remarques précieuses qui ont guidé notre travail tout au long de notre stage au CERIST.*

Je remercie également le juré composé de ***Mr. HAMMAL Youcef*** *et* ***Mr. KHENNAK Ilyes*** *pour nous avoir honorer de juger notre projet.*

Je tiens à exprimer toute ma reconnaissance à mes deux chers parents, ma petite soeur ***Amel****, et mes grand-parents pour leur soutien inconditionnel durant tout mon parcours académique.*

J'exprime toute ma gratitude à ceux qui ont revu notre mémoire et corrigé les fautes, incluant ***Mme. Souhila****,* ***Mlle Ikram*** *ainsi que mes amis* ***Nesrine*** *et* ***Djalil****.*

Je remercie spécialement ***Dr. BOUCENNA Fateh****, qui fut le responsable de notre découverte du sujet de notre PFE. Ainsi que mon oncle* ***Moulou****d qui a grandement contribué au bon déroulement du projet*

Je voudrais Pour finir, exprimer mes vifs remerciements envers mes amis ***Hocine****,* ***Ines****,* ***Chakib****,* ***Zo****,* ***Yacine****,* ***Lint****, sans oublier mon cousin* ***Raouf****, et ma binôme* ***Serine*** *qui m'ont apporté leur soutien moral et intellectuel.*

À ma très chère regrettée grand-mère

Table des matières

Table des figures

Liste des tableaux

Introduction générale

En 2019, l'Algérie comptait plus de 36000 chercheurs en activité répartis à travers 1470 laboratoires et centres de recherche au niveau national. Néanmoins le ministre de l'enseignement supérieur et de la recherche scientifique de l'époque, déclare qu' "un énorme fossé existe entre l'entreprise économique et la Recherche Scientifique ", la même source, déplore un potentiel qui reste "sous-exploité par le secteur économique et les entreprises publiques en particulier" [3]. Le potentiel humain est une ressource précieuse, qui en la gérant convenablement, va permettre une gestion plus efficace des différents projets, ce qui augmentera la productivité de l'entreprise ou autre organisme public. Cependant, afin d'exploiter cette ressource, la première étape qui est de trouver les bonnes personnes en qui poser sa confiance, reste difficile et pas très clair.

En effet, jusqu'à récemment trouver les personnes adéquates pour une tâche, nécessitait de contacter manuellement des individus, pour leur demander des suggestions, en espérant que le jugement porté sur d'autres personnes soit pertinent et justifié. Or dans le domaine académique l'activité d'un chercheur est parfois mal connu de ses pairs, ou encore qu'il existe des individus prêts à collaborer mais qu'ils ne sont pas beaucoup connus. Afin d'automatiser ce processus, et le rendre plus précis, les dernières années ont vu l'émergence de systèmes dits de recherches d'experts. Le terme expert désigne dans la littérature une personne qui possède des connaissances, compétences et de l'expérience dans un domaine donné. En Algérie, ce genre de systèmes n'est pas encore utilisé, leur adoption permettra une meilleure connexion entre le milieu scientifique et les différentes instances publiques et privées. Ainsi non seulement une entreprise privée par exemple pourra plus facilement trouver les meilleurs compétences disponibles en une technologie nouvellement lancée, mais il saura aussi possible de mieux distribuer les tâches académiques telles que la formation des jurés pour des soutenances, le sélection d'orateurs pour des conférences, ou encore trouver efficacement le personnel nécessaire à une collaboration scientifique multidisciplinaire, par une simple requête qui contient les mots clés nécessaires.

Notre projet de fin d'étude de deuxième année Master, a pour but de mettre en place un portail national de recherche d'expert, qui regroupe tous les chercheurs algériens, et qui utilisent les dernières avancées en matières de traitement automatique du langage. La recherche d'expert rentre dans le cadre de la recherche d'informations. Cette discipline est l'une des plus prolifiques en informatique. Ces derniers temps, nous avons constaté l'émergence de plusieurs modèles utilisant l'apprentissage profond, pour des tâches de traitement automatique du langage naturel tel que *Word2Vec* ou plus récemment *GPT-3*. Dès leurs apparitions ces modèles ont connu beaucoup de succès, notamment en traduction automatique. *BERT* [4] est l'un de ces modèles apparus en 2018, et mis en libre d'accès par *Google*. C'est un modèle de langue, qui utilise une structure de *transformer*. Il est utilisé dans le but de capturer la sémantique d'un mot, dans un vecteur, selon son contexte d'apparition. Cette technique est connue sous *plongement (ou embedding en anglais)* (plus de détails dans la section 2.3.2 du chapitre 2).

Dans tout système de recherche d'experts scientifiques, la 1ère étape est de représenter le profil de l'expert (son expertise), selon le type de système, cela nécessite l'obtention de données le concernant. Dans notre cas, nous avons représenté le profil de chaque chercheur, par les do-

cuments qu'il a publié. Nous nous sommes inspirés des travaux de Berger et al. [5], qui ont été l'un des premiers à utiliser le plongement avec *BERT* dans la recherche d'experts. Nous utiliserons leur meilleur résultat comme modèle de base pour les comparaisons. Dans le but de produire un système plus performant nous proposerons trois contributions principales chacune concernant une étape du processus de recherche. Premièrement, une nouvelle approche pour l'indexation, qui se base sur les phrases individuelles au lieu du document en entier, durant cette phase, nous calculerons les plongements avec deux variantes du modèle *BERT* qui sont *SciBERT* et *RoBERTa* de *Facebook*. Cela permettra de comparer l'efficacité de ces ces deux modèles. Deuxièmement, nous allons modifier la requête initialement introduite par l'utilisateur, en utilisant sa définition, et voir si cela améliore le résultat final. Cette technique est dite augmentation de la requête. Troisièmement, nous proposerons une nouvelle manière de distribuer un score sur les auteurs d'un même document, dans le but de limiter l'influence des auteurs très prolifiques. Les tests avec le modèle de base se feront dans un premier lieu sur la base de données publiées en libre accès par les auteurs : Arxiv+MAG. Certaines limitations ont été constaté dans cette base de données. Cette raison, nous a poussé à créer un nouveau dataset en utilisant la technique de *web scraping* depuis la bibliothèque ACM. Cette nouvelle base de données sera plus représentative de l'expertise des auteurs. Nous la rendrons publique pour de futurs recherches. Durant le calcul des plongements de *SciBERT*[6] et *RoBERTa*[7], nous avons constaté qu'utiliser la technologie *CUDA* de *Nvidia* pour le calcul parallèle, apporte une accélération remarquable par rapport à la CPU.

Dans le chapitre 1 nous présenterons notre organisme d'accueil qui est le **CERIST**, le chapitre 2 présentera les notions de bases de la recherche d'information (RI), nous évoquerons aussi l'utilisation des réseaux de neurones en RI, et présenterons l'état de l'art de la recherche d'experts. Dans le chapitre 3, nous détaillerons nos différentes contributions, nous illustrerons nos concepts avec des graphiques pour une meilleure compréhension. Puis, dans le chapitre suivant, nous présenterons les détails sur la base de données que nous avons créée et nous définirons la manière dont nous avons évalué notre système. L'analyse des résultats obtenus ainsi que l'interprétation se trouvent dans le même chapitre. Le chapitre final sera consacré au portail national proposé pour la recherche d'experts en Algérie. Nous conclurons notre mémoire par une conclusion générale ou nous présenterons quelques perspectives de futurs travaux.

Chapitre 1

Organisme d'accueil

1.1 Introduction

Un centre de recherche est une institution, un laboratoire ou une organisation de recherche et d'enseignement spécialisée dans un domaine spécifique, que ce soit dans le domaine de la recherche scientifique, de la recherche historique ou dans le domaine de la sociologie et des sciences sociales. L'un des centres les plus importants en Algérie est le CERIST, qui est un établissement public à caractère scientifique et technologique à vocation intersectorielle.

Dans ce chapitre, nous présentons, en détail, le CERIST, ses missions, ses services, ses divisions de recherche et enfin sa structure.

1.2 Historique du CERIST

CERIST est le centre de recherche sur l'information scientifique et technique, créé en 1985 et placé sous la tutelle du Premier ministre, sa mission principale est d'effectuer toutes les recherches liées à la création, la mise en œuvre et au développement d'un système national d'information scientifique et technique. Ensuite, il a été rattaché au Haut Commissariat à la Recherche. Enfin, le CERIST a été déclaré établissement public à caractère scientifique et technologique à vocation intersectoriel et placé sous la tutelle du ministre de l'Enseignement supérieur et de la Recherche scientifique le 1^{er} décembre 2003. Son organisation interne a été fixée et modifiée par le décret du 2 septembre 2006 [8].

En effet, le CERIST est organisé en départements administratifs et techniques et en divisions de recherche. En plus du siège central situé à Alger, le centre dispose de sites régionaux et de bureaux de liaison répartis géographiquement dans trois grands pôles du territoire (Béjaïa, Oran, Ouargla, Constantine, Sétif et Tizi Ouzou).

FIGURE 1.1 – Logo du CERIST

1.3 Missions du CERIST

Le Centre de recherche en information scientifique et technique est un organisme responsable de la réalisation des activités de recherche et de développement dans le domaine de l'information scientifique et technique.

Le CERIST a pour rôle aussi de :

— Mener toute activité de recherche relative à la création, la mise en place et le développement du système national d'information scientifique et technique.
— Promouvoir la recherche dans les domaines des sciences et des technologies de l'information et de la communication et participer à leur développement.
— Contribuer à la coordination et à la mise en œuvre des programmes nationaux d'information scientifique et technique dans un cadre concerté et en liaison avec les secteurs concernés.
— Contribuer à l'édification et à la promotion de la société de l'information par la mise en place et le développement de réseaux sectoriels d'information thématiques, notamment le réseau académique et de recherche, et d'assurer leur connexion avec les réseaux similaires à l'étranger ainsi que par le développement et la généralisation des techniques d'information et de communication dans les activités d'enseignement supérieur.
— Participer à la modernisation du système documentaire universitaire national par la mise en place notamment de bibliothèques virtuelles.
— Réunir les éléments nécessaires à la constitution de bases de données nationales dans les domaines des sciences et de la technologie et en assurer la diffusion.
— Promouvoir la recherche en matière de sécurité de l'information et des réseaux[9].

1.4 Produits/Services phares du CERIST

1.4.1 PNST

Le Portail National de Signalement des Thèses fournit l'accès aux méta-données concernant la production scientifique nationale dans le domaine des thèses (magistère et doctorat).

1.4.2 ARN

Le réseau ARN assure la prise en charge des besoins en matière d'infrastructure de réseau d'information spécialisée, via les services produits développés par le CERIST. Il regroupe l'ensemble des institutions scientifiques et technologiques (universités et centres universitaires, écoles nationales et préparatoires, centres et unités de recherche, institutions scientifiques hors secteur MESRS) et forme un réseau sectoriel national de recherche, interconnecté avec les réseaux de recherche étrangers et l'Internet.

1.4.3 ASJP

La plateforme des revues scientifiques algériennes est une plateforme de publication électronique développée et gérée par le CERIST.

Parmi les domaines qu'elle couvre, nous citons :

— Affaires, gestion et comptabilité.
— Sciences sociales.
— Pharmacologie, Toxicologie et Pharmaceutique.
— Physique et Astronomie.

1.4.4 SNDL

Le système national de documentation en ligne donne accès à une documentation électronique nationale et internationale diversifiée et très riche, touchant tous les domaines de l'éducation et de la recherche scientifique.

On distingue deux catégories d'accès à la base de données :

1. La première catégorie est accessible sans restriction à tous les étudiants, enseignants-chercheurs et chercheurs permanents au sein des campus universitaires et des centres de recherche.
2. La deuxième catégorie, en revanche, concerne l'aspect recherche, son accès se fait sans restriction du lieu de connexion, mais exige l'obtention d'un compte individuel.

1.4.5 SYNGEB

Système Normalisé de Gestion de Bibliothèques est un logiciel complet et évolutif de gestion de tous types de documents (livres, périodiques, thèses, articles et non-livres).

Il est développé par le CERIST dans le but de :

— Informatiser les bibliothèques algériennes.
— Faciliter les échanges entre les bibliothèques dans le cadre des catalogues collectifs nationaux.
— Permettre aux bibliothèques de rendre leurs catalogues visibles sur le Net.

1.4.6 CERIST WebTV

La WebTV du CERIST est une vidéothèque mise à la disposition de la communauté universitaire. Cet espace regroupe des films scientifiques, des courts reportages ainsi que des cours et des séminaires réalisés par le CERIST.

1.5 Les divisions de recherche du CERIST

Le CERIST est constitué de cinq divisions de recherche, qui sont les suivantes :

1.5.1 La Division de recherche et développement en Théories et Ingénierie des Systèmes Informatiques (DTISI)

Actuellement, cette division travaille sur des projets de R&D incluant des paradigmes informatiques émergents tels que les Big Data et les Big Graphs, le calcul parallèle et distribué sur l'infrastructure HPC [1], IoT [2] et d'autres. Elle examine aussi comment ils pourraient être appliqués aux défis du monde réel. Cette division soutient également diverses activités qui découlent des travaux de R&D, comme l'engagement dans le transfert de technologie vers d'autres secteurs par le biais de la formation (post-graduation spécialisée, stages, formation continue), des services (expertise et conseil), de la sensibilisation et de la vulgarisation, etc[10].

1. Le calcul haute performance permet de traiter les données et d'effectuer des calculs complexes à des vitesses élevées

2. L'Internet des objets est l'interconnexion entre l'Internet et des objets, des lieux et des environnements physiques

1.5.2 Division Sécurité Informatique

L'objectif majeur de cette division est d'acquérir des compétences et de développer des solutions pour assurer le bon fonctionnement des systèmes informatiques ainsi que leur sécurité contre les intrusions. En menant des actions de recherche et de développement ainsi que de formation à et par la recherche, le pôle s'efforce de contribuer au développement du réseau national d'information scientifique et technique. Les équipes de la division se concentrent sur les systèmes, les réseaux, les données et les applications à tous les niveaux[11].

1.5.3 Division Systèmes d'Information et Systèmes Multimédia

Cette division est responsable de la recherche et du développement sur les systèmes d'information et les systèmes multimédia. Elle est en charge des initiatives de recherche impliquant la modélisation de l'information, la structure de l'information, les méthodologies de gestion de l'information et les ressources de gestion de l'information (recherche d'information, stockage, etc.). Elle est en charge de la conduite de projets informatiques intégrant l'information multimédia afin de rassembler, structurer, gérer, distribuer et accéder à l'information multimédia dans les systèmes d'information du pôle développement. Elle travaille également sur des projets de conception et de mise en œuvre de systèmes d'information décisionnels pour les entreprises, ainsi que sur des études telles que des schémas directeurs informatiques, des cahiers des charges, des audits informatiques et le développement de logiciels spécifiques, en utilisant les nouvelles technologies des sciences de l'information et de la gestion[12].

1.5.4 Division Réseaux et systèmes distribués

Les activités de cette division visent à étudier, maîtriser et introduire les technologies d'e-infrastructure. Ces e-infrastructures conduisent à la maîtrise des technologies de support et à la création de nouvelles architectures applicatives telles que les architectures CLOUD et les technologies GRID pour le partage des ressources de calcul et de stockage. Ces efforts visent, d'une part, à construire des e-infrastructures telles que le réseau national académique et de recherche ARN et la grille nationale de calcul scientifique DZ e-Science GRID, et d'autre part, à consolider les compétences et le savoir-faire technologique acquis[13].

1.5.5 Division Recherche et Développement en Sciences de l'Information et Humanités Numériques (DRDHN)

La Division Sciences de l'Information et Humanités Numériques développe des outils d'analyse et d'interprétation de grands corpus multilingues à l'aide de techniques d'intelligence artificielle et de traitement automatique du langage naturel, propose des modèles mathématiques d'évaluation de l'information et de modélisation des usages numériques, présente une approche nationale du suivi stratégique des IST [3], et réfléchit au libre accès à l'information et aux données, aux nouveaux modèles économiques et au cadre juridique des IST[14]. Notre travail fait partie de cette division.

3. Information scientifique et technique

1.6 Organigramme et structure du CERIST

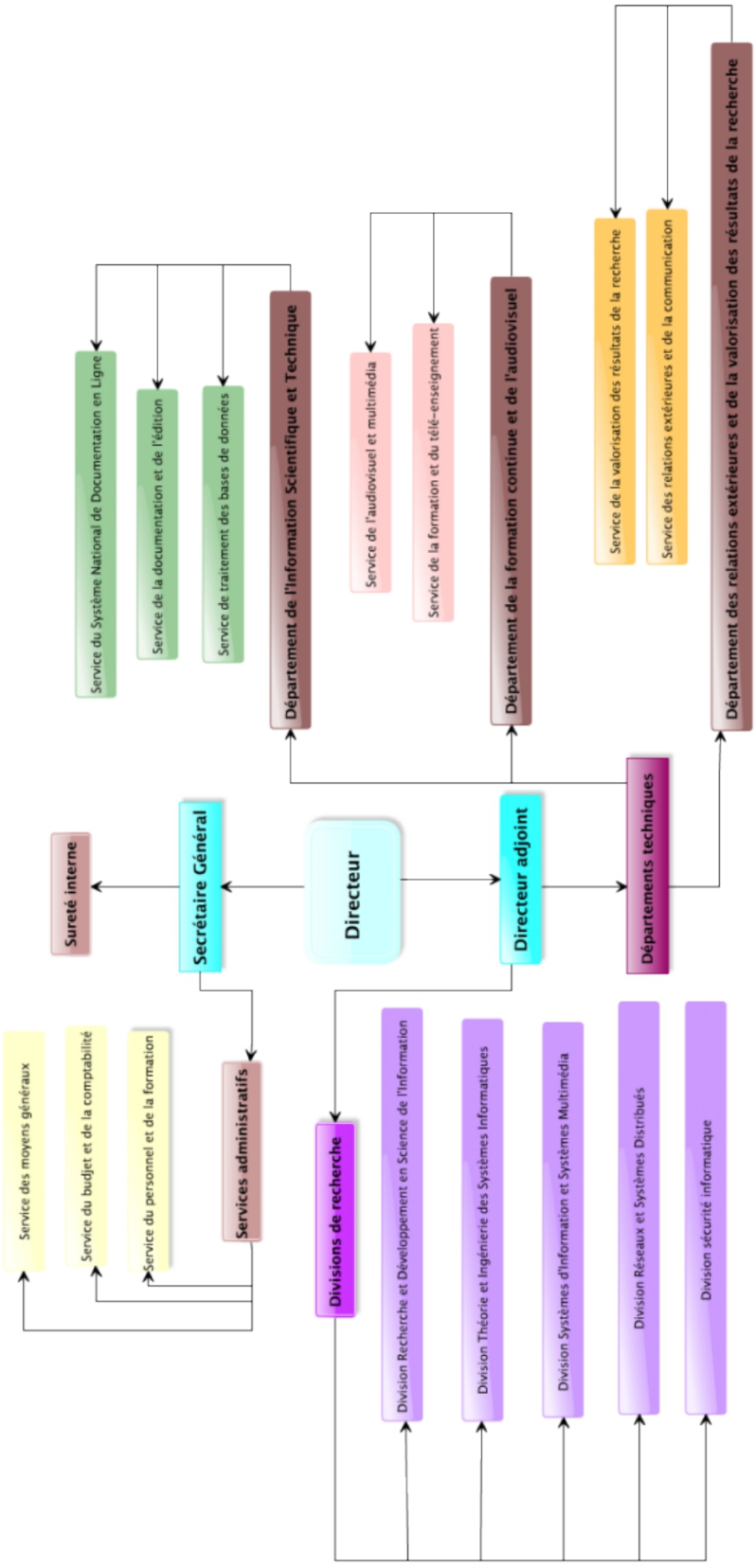


FIGURE 1.2 – Organigramme du CERIST

1.7 Conclusion

Dans ce chapitre, nous avons présenté le centre de recherche sur l'information scientifique et technique (CERIST), qui est l'organisme d'accueil.

Dans le prochain chapitre, "État de l'art", nous aborderons la recherche d'information précisément, ainsi que d'autres techniques et sujets qui s'y rapportent.

Chapitre 2

État de l'art

2.1 Introduction

Dans le deuxième chapitre du présent document, nous allons au tout début aborder les concepts de bases de la recherche d'informations (RI). La recherche d'experts y faisant partie, plusieurs de ces concepts sont présents dans la littérature. Une partie dédiée aux méthodes utilisées dans la RI nous permettra d'introduire les concepts de vecteurs et de similarité, en plus du concept de plongement, que nous détaillerons plus lorsque nous parlerons des réseaux de neurones dans la RI, nous présenterons aussi dans cette partie une classification des modèles existants. Après avoir défini certaines des métriques d'évaluations utilisées dans notre travail, nous aborderons le concept d'expansion de la requête, son utilité, et son état de l'art. Comme mentionné précédemment une section dédiée à l'utilisation des réseaux de neurones dans la RI est présente, où nous allons nous concentrer surtout sur les différentes manières d'apprendre de bonnes représentations sémantiques des termes et des documents, nous aborderons par ailleurs les différents modèles de langue utilisés dans notre projet. Enfin, la dernière section de ce chapitre présentera la recherche d'experts. Nous y mentionnerons les secteurs où elle est le plus présente ainsi que les facteurs qui la différencie des autres tâches de la RI. Nous parlerons aussi des modèles utilisés, ainsi que les bases de données disponibles.

2.2 Recherche d'information (RI)

La recherche d'information (RI) est généralement définie comme un sous-domaine de l'informatique qui s'occupe de l'organisation, le stockage, l'analyse et l'extraction de l'information [15, 16]. Dans le cas où l'information en question est textuelle, nous nous intéressons à trouver la meilleure correspondance entre une requête donnée par l'utilisateur et des documents disponibles dans la base de données.
Le classement des documents d'un corpus vis à vis d'une requête, se fait en tois étapes principales : représentation d'une requête , représention des documents [17], appariement entre requête et documents [2].

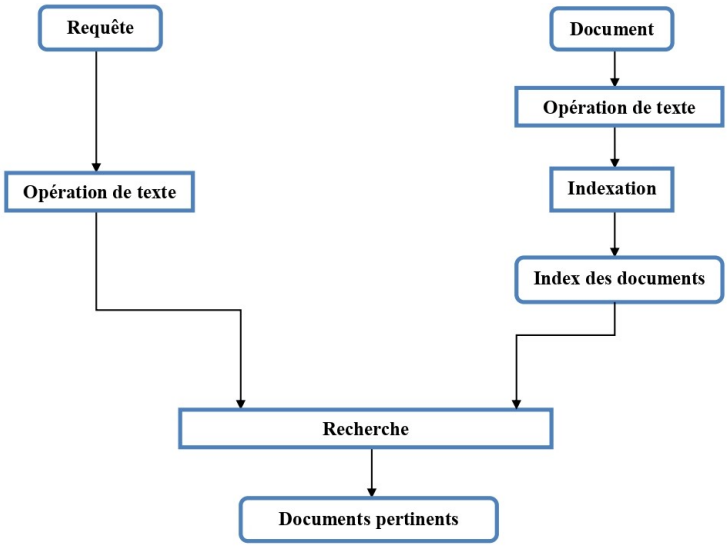


FIGURE 2.1 – Le modèle "U" des systèmes de recherche d'information

La majorité des systèmes de RI attribuent un score numérique à chaque document et classent les documents en fonction de ce score [18]. Plusieurs modèles ont été proposés pour ce processus. Les trois modèles les plus utilisés dans la recherche en RI sont : le modèle d'espace vectoriel, le modèle probabiliste, et le modèle de réseau d'inférence.

2.2.1 Les modèles de RI

Un terme (mot) est la brique la plus fondamentale dans la recherche d'information [2]. Les modèles de RI peuvent être catégorisés selon leurs dépendance ou non par rapport au terme, (également appelé propriété du modèle). Certains modèles considèrent les termes comme indépendants les uns des autres. D'autres considèrent l'interdépendance des termes et se divisent en deux catégories : ceux qui infèrent cette interdépendance, ce qui implique que c'est au modèle de la définir, et qu'elle découle de la représention de la donnée, c'est le cas par exemple de la réduction dimensionnelle (*eg : Analyse sémantique latente*), qui se base sur la matrice des occurrences (plus de détails plus loin). La deuxième catégorie de modèles s'appuie sur une source externe ou sur d'autres algorithmes pour définir cette interdépendance, l'extraction basée sur les ensembles flous en est un exemple [19].

Les modèles de RI peuvent aussi être catégorisés selon leurs bases mathématiques. De ce fait, nous notons l'existence de modèles basés sur la théorie des ensembles [20], les modèles probabilistes ainsi que les modèles algébriques [1]. La Figure 2.2 présente la relation entre les modèles de RI les plus communs, ainsi que leur classification selon leurs bases mathématiques et leurs propriétés. Le reste de cette section présentera en détails quelques-uns de ces modèles.

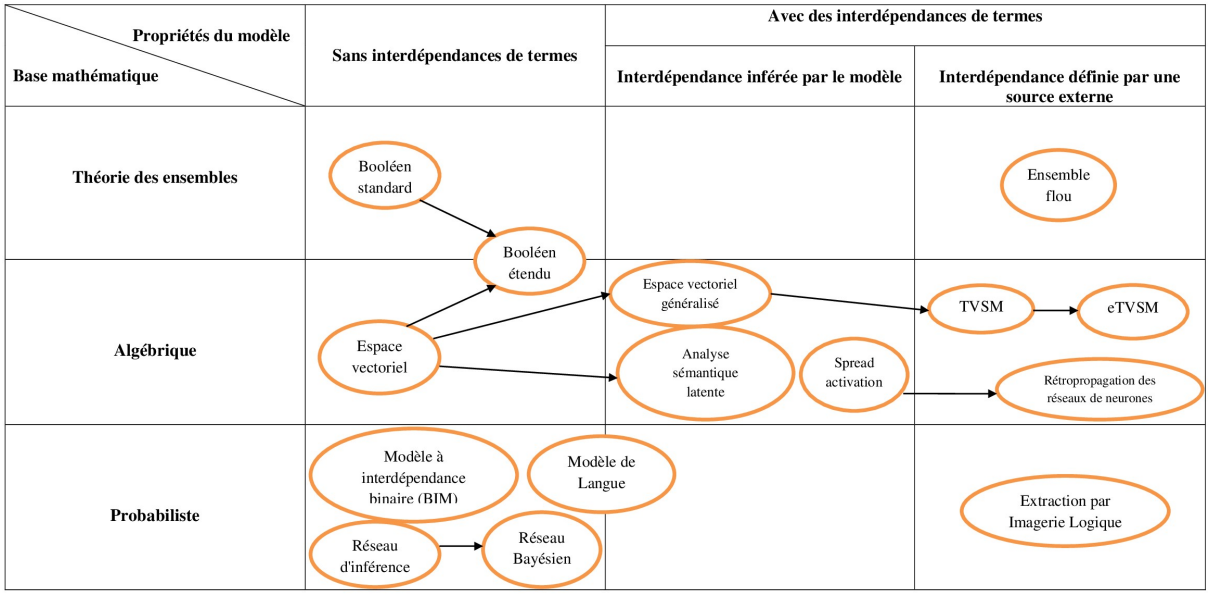


FIGURE 2.2 – Catégorisation des modèles de RI [1]

2.2.1.1 Modèle booléen

Le modèle booléen est un modèle classique de la recherche d'information [21]. Ce modèle est basé sur la théorie des ensembles ainsi que sur la logique booléenne.

Ayant un ensemble de documents (corpus), soient :

T l'ensemble du vocabulaire du corpus (l'ensemble de tous les mots) :

$$T : \{t_1, t_2, t_3, ..., t_n\}$$

l'ensemble des Documents D :

$$D : \{d_1, d_2, d_3, ..., d_n\}$$

Dans le modèle booléen, un document di, est une instanciation de T,et une requête q, est un ensemble de termes (mots) séparés par des opérateurs logiques : ET logique, OU logique ainsi que la Négation. La phase d'appariement consiste à chercher les documents qui satisfont la requête,

2.2.1.2 Modèle vectoriel

Le modèle vectoriel est un modèle algébrique de représentation d'un document [22]. Ayant un ensemble de documents (corpus), soit T l'ensemble du vocabulaire du corpus (l'ensemble de tous les mots) et N le nombre total de ces mots.

$$T : t_i, i < N$$

Chaque document d du corpus aura un vecteur v qui le représente. De sorte que l'élément vi correspond au poids qu'on attribue au terme ti pour d. Afin de définir cette pondération nous pouvons considérer le *fréquence du terme*, aussi connue sous TF (*term frequency*). Le tableau 2.1 représentent des formules utilisées pour calculer la fréquence du terme (TF) :

Schéma de pondération	Formule du TF
binaire	0, 1
fréquence brute du terme	$f_{t,d}$
normalisation logarithmique	$log(1+f_{t,d})$
normalisation $\ll 0.5 \gg$	$0.5+0.5\cdot\frac{f_{t,d}}{max_{\{t'\in d\}}f_{t',d}}$
normalisation par le max	$K+(1-K)\frac{f_{t,d}}{max_{\{t'\in d\}}f_{t',d}}$

TABLE 2.1 – Variantes de TF

Où :
$f_{t,d}$: est la fréquence du terme dans un document.
K : un pourcentage ($K \in [0,1]$)

Une manière plus courante d'attribuer ces poids est connue sous TF-IDF, qui en plus de la fréquence du terme (TF), considère aussi, la *fréquence inverse de document* IDF (*inverse document frequency*), le but étant de mesurer l'importance d'un terme pour un corpus, ainsi les termes qui se répètent le moins sont considérés comme étant ceux qui caractérisent le mieux le document.

$$idf_{i,j} = log\frac{|D|}{|d_j : t_i \in d_j|}$$

où :
log : logarithme de base 2 ou 10.
$|D|$: nombre total de documents dans le corpus
$|d_j : t_i \in d_j|$: nombre de documents où le terme t_i apparaît (c'est-à-dire $n_{i,j} \neq 0$)

Finalement le calcul de TF-IDF revient a la multiplication des deux mesures :

$$tfidf_{i,j} = tf_{i,j} \cdot idf_i$$

Une requête q introduite par l'utilisateur sera elle aussi transformée en un vecteur v_q de même schéma que celui des documents, en d'autres termes v_q est un vecteur du même espace vectoriel que celui des documents, nous pouvons donc appliquer des mesures de similarité, telles que la similarité cosinus [23] :

$$\cos\theta = \frac{A\cdot B}{\|A\|\|B\|}$$

Où :
A et B : deux vecteurs.
θ : l'angle entre eux.

Bien que simple à implémenter et assez intuitif, le modèle vectoriel présente certaines limitations. En particulier, dans sa version la plus simple, 'il ne considère pas l'interdépendance des termes dans le document ni la prise en charge des synonymes, qui est souvent une cause de

résultats incohérents dans les systèmes de RI [24]. Une autre problématique de taille est qu'un vecteur (qui représente un document) a comme dimension l'ensemble du vocabulaire du corpus ce qui le rend très difficile d'utilisation dans des cas pratiques [2].

2.2.1.3 Analyse sémantique latente (*LSI*)

Afin de pallier le problème de grande dimension du modèle vectoriel, nous pouvons utiliser des techniques de réduction dimensionnelle dont l'analyse sémantique latente fait partie. *LSI* est une méthode algébrique, à la différence des modèles vectoriels, elle prend en charge les synonymes [25], et est plus robuste face aux erreurs d'écriture des mots [26]. *LSI* utilise la matrice des occurrences, une matrice creuse, également connue comme la *matrice terme-documents*. Les lignes de cette matrice représentent les mots (termes), et les colonnes représentent les documents, les valeurs sont généralement les poids *tf-idf*.
LSI consiste en l'application d'une décomposition en valeurs singulières (*SVD*) [27] sur une matrice des occurrences X, dans le but d'obtenir sa représentation de rang inférieur[28]. Ceci revient à la résolution de $X = U.\Sigma.V^T$

$$\underset{(\vec{t}_i^{\intercal})\rightarrow}{\overset{\underset{(\vec{d}_j)\downarrow}{X}}{\begin{bmatrix} x_{1,1} & \dots & x_{1,|D|} \\ \vdots & \ddots & \vdots \\ x_{|T|,1} & \dots & x_{|T|,|D|} \end{bmatrix}}} = \underset{(\vec{t}_i^{\intercal})\rightarrow}{\begin{bmatrix} \begin{bmatrix} \\ \vec{u}_1 \\ \\ \end{bmatrix} & \dots & \begin{bmatrix} \\ \vec{u}_l \\ \\ \end{bmatrix} \end{bmatrix}} \overset{U}{\cdot} \overset{\Sigma}{\begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix}} \cdot \overset{\underset{(\vec{d}_j)\downarrow}{V^{\intercal}}}{\begin{bmatrix} \begin{bmatrix} \vec{v}_1 \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \vec{v}_l \end{bmatrix} \end{bmatrix}}$$

ou σ_1,...,σ_1 sont les valeurs singulières, et les $\vec{u}_1$,...,$\vec{u}_l$, $\vec{v}_1$,...,$\vec{v}_l$ sont les vecteurs singuliers gauches et droits respectivement.

2.2.1.4 Modèle de langue

Un modèle de langue est une distribution de probabilités sur une séquence de mots [29].Le modèle de langue utilisé en recherche d'information, est connu sous le modèle vraisemblance avec la requête (*Query likelihood model*) [30]. Ce modèle construit une distribution (ou modèle de langue) pour chaque document de la collection, pour ensuite calculer une probabilité pour chaque document ayant une requête [31]. Pour ceci nous appliquons le théorème de bayes pour calculer un score pour chaque document de la collection, ou en d'autres termes la probabilité d'un document ayant une requête $p(d/q)$ (d : document, q : requête) :

$$p(d/q) = \frac{p(q/d) * p(d)}{p(q)}$$

probabilité de la requête $p(q)$ peut être simplifiée, car la requête est la même pour tous les documents :

$$p(d/q) = p(q/d) * p(d)$$

$p(d)$ est la probabilité a priori d'un document, et peut par exemple prendre une valeur décrivant la qualité d'un document comme *pageRank*, mais est généralement ignorée, et nous supposons que tous les documents ont la même probabilité a priori :

$$p(d/q) = p(q/d)$$

C'est donc $p(q/d)$ qui est la vraisemblance d'une requête pour un document, qui est calculée pour classer les documents.

$$p(d/q) = p(q/d) = p(q/M_d)$$

ou M_d est le modèle de langue du document d :

$$p(q/M_d) = \prod p(t/d)$$

ou t sont les termes de la requête, supposés sans relations entre eux.
$p(t/d)$ est la probabilité du terme t dans le document d, nous utilisons généralement la fréquence du terme (tf) pour la calculer.

$$p(q/M_d) = \prod p(t/d) = \prod \frac{tf(t,d)}{taille(d)}$$

Le problème avec cette formulation est que si au moins un terme de la requête n'apparaît pas dans le document, alors le modèle ne fonctionnera pas. Pour y remédier, nous utilisons des méthodes de *smoothing*, dont les plus connues sont :

1. Jelinek-Mercer [32] :

$$p(t_q|d) = \left(\lambda \frac{tf(t_q,d)}{|d|} + (1-\lambda)\frac{\sum_{\overline{d}\in D} tf(t_q,\overline{d})}{\sum_{\overline{d}\in D}|\overline{d}|}\right)$$

2. Dirichlet Prior Smoothing [33] :

$$p(t_q|d) = \left(tf(t_q,d) + \mu \frac{\sum_{\overline{d}\in D} tf(t_q,\overline{d})}{\sum_{\overline{d}\in D}|\overline{d}|}\right) / \left(|d| + \mu\right)$$

2.2.2 Expansion de la requête

L'expansion de la requête (QE) est une technique utilisée dans le processus de recherche d'information. Elle a pour but de réduire l'écart entre la requête et le document, afin d'améliorer la qualité des résultats[34].

La formulation originale de la requête peut donner des résultats moins pertinents par rapport à une requête étendue, dus à plusieurs raisons, parmi lesquelles, la requête originale est très courte pour capturer le sens réel et complet (la taille moyenne d'une requête sur le web est de 2,4 mots [35, 36]), une autre raison, tout aussi valable, est que parfois l'utilisateur ne connaît pas les résultats qui l'intéressent avant à l'introduction de la requête, aussi il est possible que les termes de la requête sont en relation avec différents sujets[37].

Le problème de vocabulaire [24] est le résultat d'une combinaison de synonymie (mots ayant un sens commun) et de polysémie (mots ayant plusieurs sens). Ce problème mène à une réduction des taux de rappel et de précision. De nombreuses techniques ont été conçues pour traiter le problème du vocabulaire, comme le retour de pertinence, le filtrage interactif des requêtes, les modèles de connaissances dépendants du corpus, les modèles de connaissances indépendants du corpus, le regroupement des résultats de recherche et la désambiguïsation du sens des mots[37].

Les premiers travaux dans ce domaine remontent jusqu'à 1960[38]. La plupart des contributions ont utilisé l'extension de requête en ajoutant des termes qui ont une relation sémantique avec la requête [37]. Par exemple, Saar et al. [39] ont utilisé le plongement de mots (Word2Vec's CBOW) pour trouver les termes sémantiquement liés à la requête. Un autre travail a utilisé une métaheuristique (Bat algoritm) afin de trouver les meilleurs termes pour étendre la requête [40].

Les modèles d'expansion de requêtes peuvent être classés en trois catégories : manuel, automatique et interactif. L'expansion manuelle des requêtes repose sur les connaissances et l'expérience du chercheur pour sélectionner les termes appropriés à ajouter à la requête. L'expansion automatique des requêtes pondère les termes candidats à l'expansion en traitant les documents retournés lors du premier passage de la recherche, et étend la requête originale en conséquence. L'extension interactive des requêtes automatise le processus de pondération des termes, mais c'est l'utilisateur qui décide des termes à étendre[41]

Les méthodes modernes d'expansion de requêtes impliquent une analyse de la collection de documents ou sont basées sur des dictionnaires ou des ontologies. Une autre direction dans l'expansion de requête est l'application de plongement de mots.

2.2.3 Métriques d'évaluation

Le domaine de la recherche d'information a utilisé divers types de mesures quantitatives pour évaluer si les résultats de la recherche répondent à l'intention de l'utilisateur.Dans ce qui suit, nous présentons ces mesures.

2.2.3.1 La précision et le rappel (*Precision and recall*)

La précision est la fraction des documents pertinents par rapport au nombre total de documents dans l'ensemble extrait Rq, tandis que le rappel est la fraction des documents pertinents par rapport au nombre total de documents pertinents dans la collection D. Les deux supposent que les étiquettes de pertinence sont binaires.Ils sont calculée par les fonctions suivantes :

$$Precision_q = \frac{\sum_{\langle i,d\rangle \in R_q} rel_q(d)}{|R_q|}$$

$$Recall_q = \frac{\sum_{\langle i,d\rangle \in R_q} rel_q(d)}{sum_{d\in D} rel_q(d)}$$

Où :
q : la requête
Rq : le nombre total de documents dans l'ensemble récupéré
D : le nombre total de documents pertinents dans la collection

2.2.3.2 La moyenne de la précision moyenne (*Mean average precision*) (MAP)

moyenne est une mesure qui combine le rappel et la précision pour les résultats de recherche classés. La moyenne de cette mesure sur toutes les requêtes donne le score MAP pour l'ensemble des requêtes.

$$AveP_q = \frac{\sum_{\langle i,d\rangle \in R_q} Precision_{q,i} x rel_q(d)}{\sum_{d\in D} rel_q(d)}$$

Où :
$Precision_{q,i}$: est la précision calculée au rang i pour la requête q.

2.2.3.3 Le rang moyen réciproque (*Mean reciprocal rank*) (MRR)

Le rang réciproque moyen est un score relatif qui calcule la moyenne de l'inverse des rangs auxquels le premier document pertinent a été retrouvé pour un ensemble de requêtes.

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

Où :
$rank_i$: est la position du résultat pertinent dans la i'ième requête.
Q : est le nombre total de requêtes.

2.2.3.4 Le gain cumulatif actualisé normalisé (*Normalized discounted cumulative gain*) (NDCG)

Le gain cumulatif actualisé normalisé est une mesure de la qualité du classement. C'est la somme des vrais scores classés dans l'ordre induit par les scores prédits, après application d'une décote logarithmique, puis la division par le meilleur score possible (DCG idéal, obtenu pour un classement parfait).

$$DCG_q = \sum_{\langle i,d\rangle \in R_q}\frac{2^{rel_q(d)}-1}{log_2(i+1)}$$

$$NDCG_q = \frac{DCG_q}{IDCG_q}$$

Le DCG idéal ($IDCG_q$) est calculé de la même manière, mais en supposant un ordre de classement idéal pour les documents jusqu'au rang k.

2.3 Réseaux de neurones (RN) dans la RI

Comme mentionné précédemment les trois étapes principales que suivent les systèmes de recherche d'informations afin de classer les documents d'une base de données sont la représentation d'une requête, la représentation des documents et appariement entre requête et documents. Les réseaux de neurones peuvent intervenir dans une ou plusieurs de ces étapes. Par exemple certains systèmes utilisent l'apprentissage profond pour une meilleure représentation de la requête [42], les auteurs de la référence [17] proposent une une manière d'apprendre une représentation efficace des données textuelles en utilisant les réseaux de neurones, ces derniers ont aussi été utilisé au niveau de l'appariement (matching) entre requête et document [43, 44]. Dans les sections suivantes, nous présenterons les notions de base des réseaux de neurones, puis nous nous intéresserons à l'aspect de représentation des données, avant de voir les différentes manières d'utiliser l'apprentissage profond pour transformer du texte en forme vectoriel, ainsi que les avantages de cette pratique.

2.3.1 Définitions

Un réseau de neurones artificiels ou plus couramment réseau de neurones (RN), est une structure mathématique inspirée par la biologie [45]. C'est une collection de noeuds connectés aussi appelés neurones artificiels. Un neurone artificiel, parfois appelé neurone formel est la structure de base d'un réseau de neurones. Le premier neurone artificiel proposé est celui de de McCulloch-Pitts [46]. Mathématiquement un neurone artificiel est une fonction a m variables

(signaux ou encore stimuli) a valeurs réelles. Dans le modèle de McCulloch-Pitts, chaque variable *xi* est associée à une valeur numérique appelé poids synaptique w_i, la première opération du neurone est d'appliquer la somme pondérée aux entrées reçues (les variables *xi*) en plus d'une autre grandeur qu'on appelle le biais (ou seuil) b qui rendra la fonction (neurone artificiel) plus flexible. :

$$S = w_0x_0 + w_1x_1 + w_2x_2 + w_3x_3 + b = \sum_{i=1}^{m} w_ix_i + b$$

La somme S précédente sera passée à une fonction dite d'activation, dans la formulation originale de McCulloch-Pitts, c'est la fonction de Heaviside (fonction marche d'escalier) [47], qui prend les valeurs 0 et 1 :

$$\begin{cases} 0 & \text{si } S < b \\ 1 & \text{sinon.} \end{cases}$$

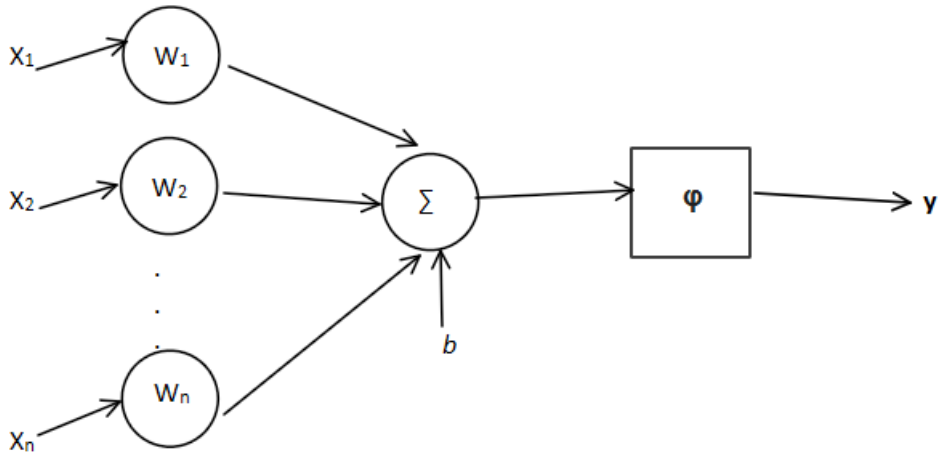


FIGURE 2.3 – Neurone artificiel

Autre la fonction escalier, il existe d'autre fonction d'activations, parmi lesquelles la fonction Unité de rectification linéaire (ReLu) [48], sigmoïde, Unité exponentielle linéaire (ELU). Tangente hyperbolique, et d'autres.

Comme mentionné précédemment un réseau de neurones est une collection de neurones formels connectés. Un réseau de neurone est composé d'une couche d'entrée, qui est par exemple un vecteur dit *one hot encoding* [1], et une de sortie ainsi qu'une ou plusieurs couches cachées, les réseaux ayants plusieurs couches cachées sont appelés réseaux de neurones profond (*DNN pour Deep Neural network en anglais*), et sont utilisés dans l'apprentissage profond (*Deep learning en anglais*) [49, 50] L'autre élément essentiel des réseaux de neurones est la fonction de perte, cette fonction calcule l'erreur entre les valeurs prédites par le réseau et les valeurs réels, le but de la phase d'apprentissage est de minimiser cette erreur. Voici les trois fonctions de pertes les plus connues :

1- *Mean Squared Error, L2 Loss* [51] : La perte MSE est utilisée pour les tâches de régression. Cette perte est calculée en prenant la moyenne des différences au carré entre les valeurs réelles (cibles) et prédites.

1. Dans l'exmple du TALN chaque terme est associé à un vecteur qui a la taille de tout le vocabulaire du corpus, chaque position de ce vecteur correspond a un mot du vocabulaire, c'est un vecteur binaire qui a des 0 partout à l'exception d'une seule position qui est celle du terme en question.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Où :
MSE : Erreur quadratique moyenne
n : le nombre de points de données
Y_i : valeurs observées
$\hat{Y}_i$: valeurs prédites

2- *Binary Cross Entropy* [52] : La perte BCE est la fonction de perte par défaut utilisée pour les tâches de classification binaire. Elle nécessite une couche de sortie pour classer les données en deux classes et la plage de sortie est (0-1), c'est-à-dire qu'elle doit utiliser la fonction *sigmoïde* [2].

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i.log(\hat{y}_i)$$

Où :
$\hat{y}_i$: les distributions de probabilité prédites par le classificateur pour prédire une classe.
m : le nombre d'enregistrements

3- *Categorical Cross Entropy* [53] : Il s'agit de la fonction de perte par défaut lorsque nous avons une tâche de classification multi-classes. Elle nécessite le même nombre de nœuds de sortie que les classes, et la couche finale passant par une activation softmax de sorte que chaque nœud de sortie ait une valeur de probabilité comprise entre (0-1).

$$logloss = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{M} y_{ij}.log(p_{ij})$$

Où :
N : nombre de lignes
M : nombre de classes

Le dernier aspect important des RNs est la mise a jour des différents poids, ceci est fait après avoir calculé l'erreur par la fonction de perte en utilisant la descente du gradient[54].

2.3.1.1 Architecture de *DNN* en traitement automatique du langage naturel

Il existe plusieurs types d'architectures de réseaux de neurones, chacun optimisé à un type de données et une tâche précise, par exemple en vision par ordinateur ce sont les réseaux de neurones à convolutions qui sont le plus utilisés [55]. Le traitement automatique du langage naturel (TALN) est une autre application des réseaux de neurones. Le langage étant la 1ere source de communication des humains, avoir des modèles performants en TALN, s'avère très utile en recherche d'informations. Les données utilisées en TALN sont dites séquentielles, cela veut dire que considérer l'ordre entre les différents mots est important, pour comprendre la signification d'une phrase. Les premiers modèles prenants en charge l'aspect séquentiel des données sont les réseaux de neurones récurrents (RNN). Les RNN présentent néanmoins plusieurs problèmes, entre autres, ils ne sont pas très performants lorsqu'ils travaillent avec de longues séquences de texte. Un autre problème important est qu'ils sont très longs à entraîner. En effet, le fait de

2. La fonction sigmoïde représente la fonction de répartition de la loi logistique. Elle est souvent utilisée dans les réseaux de neurones.

traiter les mots séquentiellement, les empêchent de profiter des accélérations des GPU [3]. Afin de combler ces manques, une nouvelle classe de modèles appelée *transformers* fut développée en 2017, par des chercheurs de Google et de l'université de Toronto [56]. À la différence des RNN, les *transformers* peuvent profiter du calcul parallèle des *GPU*, ce qui leur permet de s'entraîner sur de très grande quantité de données. Autre avantage est que l'entraînement ne nécessite qu'un petit nombre d'étapes définit de façon empirique. La première innovation est l'utilisation de ce que les auteurs appelle *positional-encoding*, qui est une mécanisme qui permet de connaître la position relative à chaque *tokens* (mots), ce mécanisme est très important car les *transformers* n'utilisent pas de réseaux récurrents ni de convolution pour connaître l'entourage de chaque mots (ce qui comme mentionné est important en TALN). L'autre innovation est l'application d'un mécanisme de *self-attention* dans chaque étape, ce mécanisme permet de modéliser directement la relation entre les mots d'une phrase. Dit autrement, ce mécanisme compare chaque mot m_i d'une phrase avec tout son entourage. Le résultat de cette comparaison est un score d'attention pour chaque mots entourant m_i. Ces scores une fois calculés déterminent le taux de contribution de chaque mots à la prochaine représentation de m_i, à l'étape suivante. Le mécanisme d'attention n'est pas nouveau, il est au fait apparu 2 ans avant les *transformers* en 2015 dans le cadre des traductions automatiques, ce que les *transformers* apportent c'est la possibilité d'utiliser ce mécanisme pour non seulement la traduction automatique, mais aussi pour d'autres tâches de TALN telles que compléter les parties manquantes d'un discours, comprendre les règles de la grammaire de langage, obtenir une représentation vectorielle de la sémantique d'un mot (ce que nous allons détailler plus dans le chapitre suivant). Ceci est dû à la nature même des réseaux de neurones profonds : leurs couches cachées contiennent des représentions internes des données au-quels ils sont entraînés, par exemple dans un modèle de vision par ordinateur, certaines cachées profondes peuvent se spécialiser dans la reconnaissance de contours, d'autres de lignes ...etc.

3. GPU pour Graphical Process Units, sont très utilisés lors de l'entraînement des réseaux de neurones profonds

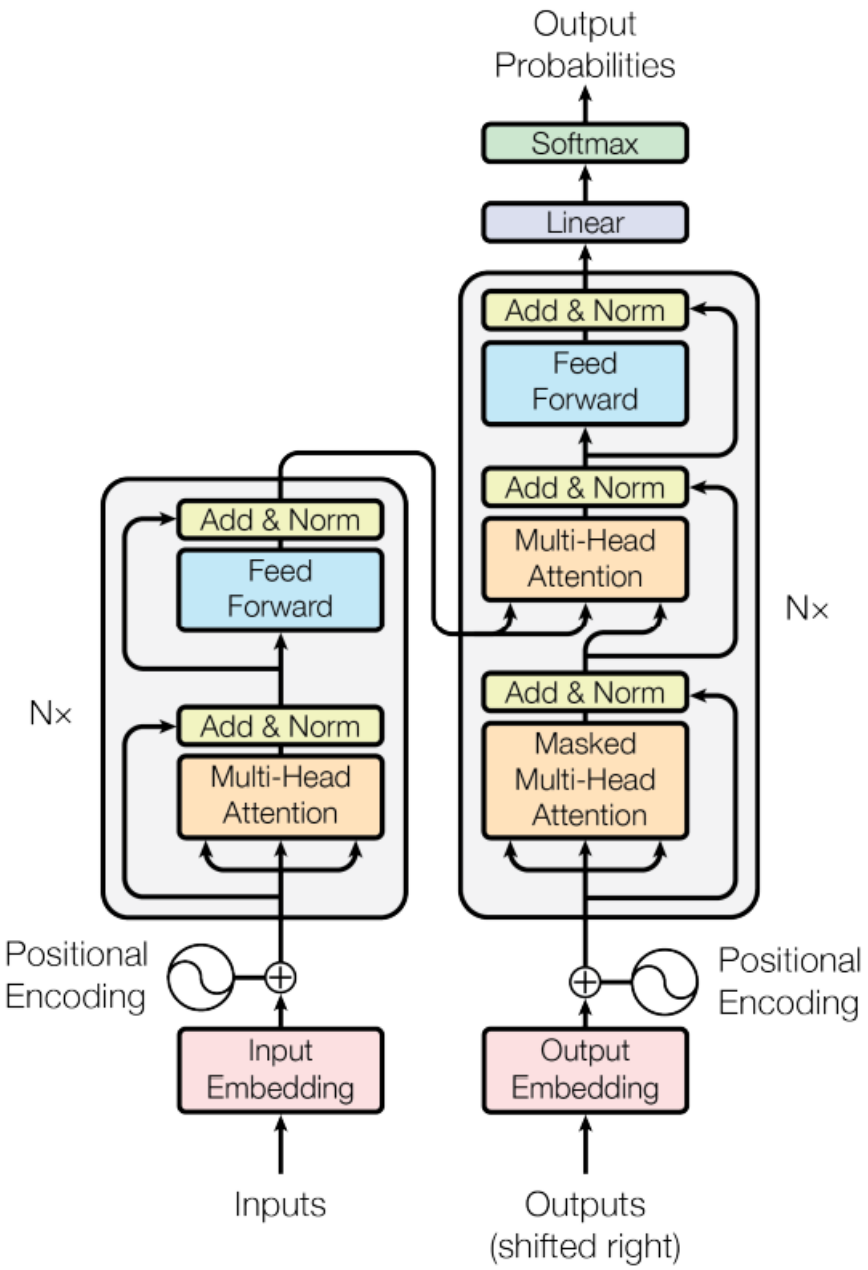


FIGURE 2.4 – architecture originale des *transformers* apparue dans l'article *Attention Is All You Need*

2.3.2 Représentation des données

La représentation des données sous forme vectorielle est très présente en recherche d'information, et constitue un élément clé des réseaux de neurones. Différentes représentations vectorielles mènent à des niveaux de généralisations différents.

2.3.2.1 Représentation distribuée

Dans cette représentation chaque terme est associé à un vecteur qui a pour but de capturer sa sémantique. Ces vecteurs peuvent être catégorisés en deux grandes familles [2], vecteurs à caractéristiques observables, et vecteurs latents.

2.3.2.1.1 Espace de caractéristiques observables

Une représentation de ce type est définit par le choix des caractéristiques, et la stratégie de pondération (TF-Idf, positive pointwise mutual information), concernant les caractéristiques observables, cela peut s'agir des documents dans lesquels un terme apparaît, comme le montre l'exemple suivant :

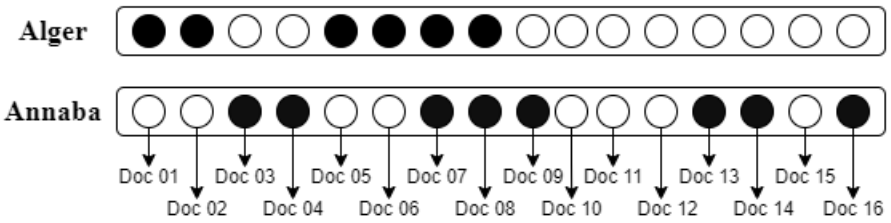


FIGURE 2.5 – Utilisation de documents comme des caractéristiques observables

Selon l'hypothèse distributionnelle [57], les termes qui apparaissent dans un même contexte, ont tendance à être sémantiquement similaires, une idée similaire a été formulé par Firth et al. [58], selon laquelle "un terme est caractérisé par la compagnie qu'il côtoie". Voici deux exemples de représentation des caractéristiques, qui en découlent de cette idée, il s'agit des caractéristiques liées aux "termes voisins", sans et avec considération de la distance.

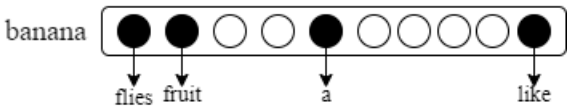


FIGURE 2.6 – Utilisation des termes voisins comme caractéristiques observables[2]

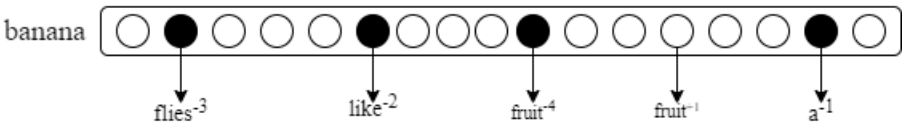


FIGURE 2.7 – Utilisation des termes voisins comme caractéristiques observables en considérant la distance[2]

Les exemples précédents, représentent des caractéristiques externes au terme, néanmoins, nous pouvons utiliser des caractéristiques interne au termes, telles que les trigraphe. [2]

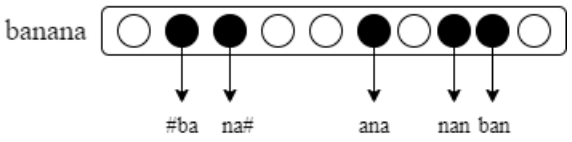


FIGURE 2.8 – Utilisation Utilisation des trigraphes comme caractéristiques observable[2]

2.3.2.1.2 Espaces de caractéristiques latentes

Le problème avec la représentation par caractéristiques observables, est que les vecteurs produits sont généralement de très grande dimensions (la taille de tout le vocabulaire par

exemple), ce qui les rend difficile, voire impossible d'utilisation dans des cas pratiques. Pour y remédier, plusieurs techniques ont vu le jour pour apprendre des représentations des vecteurs qui préservent les propriétés intéressantes offertes par la modélisation par caractéristiques observables, mais qui soit de dimension raisonnable, ce type de vecteur est également appelée *embedding*. [59]. Les méthodes qui existent pour obtenir ce plongement (*embedding*), inclut les méthodes de factorisation matricielles et les réseaux de neurones. Nous avons déjà évoqué le modèle de factorisation matricielle *LSI* dans la partie Les modèles de RI, dans ce qui suit nous nous intéresserons aux techniques de plongement basées sur les réseaux de neurones. Au lieu de factoriser la matrice des occurrences, les modèles utilisant des réseaux de neurones apprennent les plongements en mettant en place une tâche de prédiction de caractéristiques et utilisent des architectures motivées par le principe du goulot d'étranglement de l'information [2]. Dans les sections suivantes, nous présenterons quelques unes de ces approches. Après avoir obtenu le plongement d'un mot dans un espace vectoriel, deux mots similaires occuperons des positions proches les uns des autres, alors que des mots différents occuperons des positions éloignées, de cette façon nous pouvons déterminer si deux mots ont la même sémantique par des opérations d'algèbres linéaire. Nous pouvons obtenir un vecteur représentatif d'une phrase ou d'un document en effectuant la moyenne des plongements des mots de la phrase ou document, obtenus par les réseaux de neurones.

2.3.2.1.3 Word2vec

Word2vec utilise un réseau de neurones composé d'une seule couchée de plusieurs centaines de neurones, d'une couche d'entrée et une de sortie. Il existe en deux version CBOW et *Skip-gram*, dans la première, il calcule le plongement d'un mot *mi* selon les mots l'entourant, néanmoins il ne prend pas en considération l'ordre des mots autour de *mi* (hypothèse de *sac de mots*). Le modele *Skip-gram*, prend en entrée un mot et predit les mots l'entourant
Word2Vec apprend les plongements au niveau des mots, de ce fait, il ne peut générer des représentations que pour les mots qui existent dans son ensemble d'entraînement. Selon les auteurs, CBOW est plus rapide, le skip-gram est meilleur pour travailler avec des mots peu fréquents [60] .

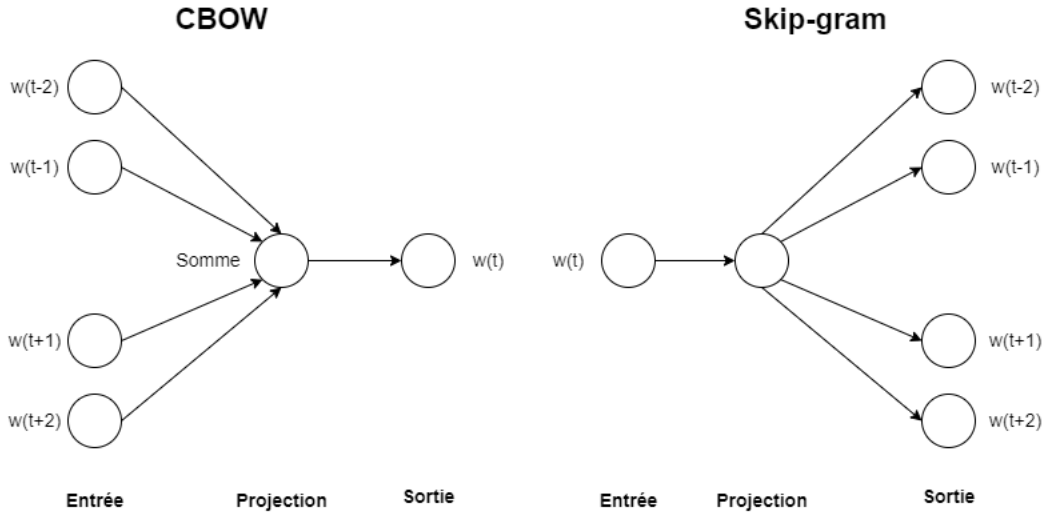


FIGURE 2.9 – L'architecture des deux modèles CBOW et Skip-gram.

2.3.2.1.4 BERT

BERT (*Bidirectional Encoder Representations from Transformers*) est un modèle de langue développé par Google AI en 2018.L'innovation clé de BERT est à d'appliquer l'entraînement bidirectionnel au Transformer. C'est-à-dire l'encodeur du Transformer dans ce modèle, lit toute la séquence de mots en une fois. Cette caractéristique permet au BERT d'apprendre le contexte d'un mot en fonction de tout son environnement (gauche et droite du mot).
L'entrainement de BERT se fait en deux phases. La première est le pré-entraînement, c'est là que BERT comprend le langage, et la seconde est l'ajustement fin, dans cette phase le modèle apprend à utiliser le langage pour une tâche spécifique.

Le but du pré-entraînement est de faire en sorte que BERT apprenne quel est le langage et quel est le contexte. Le modèle apprend la langue en s'entraînant simultanément sur deux tâches non supervisées. Il s'agit de la modélisation du langage masqué (MLM) et de la prédiction de la phrase suivante (NSP). Dans le MLM, BERT prend une phrase avec des mots aléatoires remplis de masques, l'objectif est de produire ces tokens masqués. Dans le NSP, BERT prend deux phrases et détermine si la deuxième phrase suit la première (problème de classification binaire). En utilisant à la fois le MLM et le NSP, BERT obtient une bonne compréhension du langage.

BERT peut, en peu de temps, être affiné sur une tâche spécifique en aval avec relativement peu d'étiquettes, car les modèles linguistiques généraux ont déjà été appris lors du pré-entraînement. Jacob Devlin et al [61] écrivent que l'ajustement fin de BERT est "simple", simplement en ajoutant une couche supplémentaire après la couche finale de BERT et en entraînant le réseau entier pendant quelques époques seulement.

En termes de données d'entraînement, BERT a été entraîné sur 2,5 milliards de mots de la Wikipedia anglaise et 800 millions de mots du BooksCorpus de Google, ce qui donne un ensemble de données massif de 3,3 milliards de mots.

2.3.2.1.5 RoBERTa

RoBERTa est une version de BERT pour laquelle certains hyperparamètres du pré-entraînement ont été modifiés.

Les modifications apportées par Liu et al.[7] sont simples, elles comprennent : 1) un entraînement du modèle plus long, avec des lots plus importants, sur davantage de données ; 2) la suppression de l'objectif NSP ; 3) un entraînement sur des séquences plus longues ; et 4) un changement dynamique du modèle de masquage appliqué aux données d'entraînement. Ils ont également collecté un nouvel ensemble de données volumineux, afin de mieux contrôler les effets de la taille de l'ensemble de traitement.

2.3.2.1.6 SciBERT

SciBERT est un modèle de langue pré-entraîné basé sur BERT pour répondre au manque de données scientifiques étiquetées de haute qualité et à grande échelle.Il exploite le pré-entraînement non supervisé sur un grand corpus multi-domaine de publications scientifiques (1.14 million d'articles, 3.1 milliards de mots) pour améliorer les performances sur les tâches NLP scientifiques en aval. Beltagy et al.[6] ont utilisé le texte entier des articles dans l'entraînement, et pas seulement les résumés.

SciBERT possède son propre vocabulaire (scivocab) qui est construit pour correspondre au mieux au corpus d'entraînement. Ils ont entraîné des versions avec et sans majuscules. Ils ont également inclus des modèles formés sur le vocabulaire original de BERT (basevocab) pour comparaison. Ses modèles sont :

— **scibert-scivocab-uncased** : le vocabulaire ”scivocab”, et sans majuscules.
— **scibert-scivocab-cased** : le vocabulaire ”scivocab”, et avec majuscules.
— **scibert-basevocab-uncased** : le vocabulaire ”basevocab”, et sans majuscules.
— **scibert-basevocab-cased** : le vocabulaire ”basevocab”, et avec majuscules.

2.4 La recherche d'expert

Le terme expert est généralement donné à une personne ayant des connaissances, des compétences ou une formation particulière dans un domaine donné [62]. La recherche d'experts (RE) fait référence à l'activité de sélection de personnes ayant des profils intéressants pour une tâche spécifique.

Durant les récentes années, les systèmes de recherches d'experts ont été largement utilisés dans différents domaines, tels que les communautés de questions réponses [63, 64], communautés de connaissances en ligne [65], réseaux sociaux [66] et dans le domaine académique [67, 68, 69]. Une large recherche a été menée pour déterminer les secteurs principaux ou de tels systèmes ont été utilisés [70]. Les résultats montrent que le domaine académique est celui qui a connu le plus grand nombre de travaux liés, suivi par le domaine de l'industrie [71, 72], et de la médecine [73, 74]. Les principales tâches des systèmes de recherches d'experts dans le domaine académique sont la sélection de personnes pour la révision de documents, les collaborations de recherche, les collaborations entre les universités et les acteurs industriels, et enfin la recherche d'un superviseur approprié.

2.4.1 La source de l'expertise

La source de l'expertise désigne les données qui sont utilisées dans le processus de recherche d'expert. Cette source provient en grande partie des données textuelles en relation avec l'expert, ce qui peut inclure les informations sur son profil (nombre de citations, affiliation), ses communications (e-mails, message textuel) [75], et ses documents publiés, qui est la source textuelle la plus importante [70]. Certain système de RE utilisent les réseaux sociaux comme source d'expertise [76, 77], certains ont eu recours au forum et au mots clés des questions posées par les auteurs [78, 79, 80].Plusieurs études récentes suggèrent qu'utiliser différentes sources d'expertise dans un même système, améliore la qualité des résultats obtenus [81, 82].

La recherche d'informations est un domaine très prolifique, qui utilisent les technologies les plus modernes [2] ainsi beaucoup de ses modèles ont été appliqué à la recherche d'experts, Autre point positif qui est très important porte sur la disponibilité des base de données, ceci est rendu possible entre autre grâce a l'entreprise TRECT [4] [83] (plus de détails sur les bases de données plus loin). Néanmoins certains désavantages spécifiques à la la tâche de recherche d'expert existent. En effet, il est parfois difficile pour les organisations de maintenir les données liées à son personnel à jour [70], durant notre travail nous avons aussi remarqué que souvent un auteur publie dans des revues différentes certains des ses articles de façon exclusive, ce qui complique encore plus l'appréhension de sa réelle expertise.

Certains facteurs peuvent aussi être pris en considération lors de la conception d'un système de RE. En effet, à la différence de la RI classique qui a pour but de fournir une réponse matérielle à une requête [84], le processus de RE implique la collaboration d'êtres humains. De ce fait, la connaissance à elle seule sans de bonnes relations entre les experts n'est pas suffisante [85], il se peut aussi qu'une personne ne soit pas disposée à partager ses connaissances avec d'autres, en raisons des différences entre eux [86]. Certaine études s'intéressent à d'autres aspects des experts tel que la réputation [87], le temps nécessaire à l'obtention du service [88],

4. Text REtrieval Conference

la langue et la disponibilité [89] En raison de la nature non objective de certains de ces facteurs, il est évident que c'est difficile de les mettre en place dans des cas pratiques.

2.4.2 Les modèles utilisés

Les trois composantes principales d'un système de recherche d'expert sont : les candidats (les experts), leurs sources d'expertise (discuté dans le section précédente) et les requêtes [90]. Plusieurs modèles ont été proposés, afin de capturer la relation entre les termes d'une requête et l'expert.

2.4.2.1 Les modèles probabilistes

Les modèles probabilistes se divisent en deux parties : les modèles probabilistes génératifs et discriminatifs. Après la première édition de l'entreprise TREC en 2005, plusieurs approches se basant sur des modèles probabilistes génératifs ont été proposées. L'idée de base est de classer les candidats vis-à-vis d'une requête, en estimant la probabilité *p(candidat / requête)*, en d'autre terme estimer la vraisemblances qu'un candidat soit un expert pour cette requête.
Les approches probabilistes génératives sont fondées principalement sur les modèles de langue [91]. Balog et al. [92] ont fait remarquer qu'il existe deux types de modèles génératifs : basé candidats et basé requête. Les modèles génératifs basé candidats estiment la probabilité *p(candidat / requête)* directement. A titre d'illustration, Cao et al [93] ont utilisé un modèle de langue à deux niveaux, composé d'un modèle de pertinence et d'un modèle de cooccurrence. Au lieu de calculer directement la probabilité *p(candidat / requête)* les modèles génératifs basés requête, utilisent la formule de Bayes :

$$p(candidat|requête) = \frac{p(requête|candidat)p(candidat)}{p(requête)}$$

Pour une requête donnée, p(requête), est considérée comme constante, la formule précédente peut être simplifiée en :

$$p(candidat|requête) \propto p(requête|candidat)p(candidat)$$

$p(candidat)$ est la probabilité a priori qu'un candidat soit un expert, $p(requête|candidat)$
Le modèle discriminant est l'autre catégorie de modèles probabilistes avec une base statistique solide. Ces modèles ont été utilisés dans plusieurs tâches de RI [70]. Concernant la recherche d'experts, certains travaux ont été effectués avec ce genre de modèles, mais leur nombre reste moindre par rapport aux modèles génératifs.

2.4.2.2 Les modèles de votes

Les modèles de vote utilisent des techniques de fusion de données afin d'attribuer un score pour chaque candidat, dans le but d'obtenir une liste ordonnées des meilleurs candidats pour une certaine requête [94].

Dans le contexte de recherche d'experts (RE), les publications d'une personne peuvent être considéré comme une preuve de son expertise, ainsi dans , les systèmes de RE se basant sur les modèles de votes calculent d'abord un score pour les documents d'un candidat (par rapport à une requête q), pour ensuite utiliser une formules de fusion de données, pour combiner les scores des documents, afin obtenir un score final a un candidat e. Le tableau suivant présente une liste de formule de fusion de données utilisées :

Nom	Le score de pertinence du candidat est :
Votes	$\lvert M(e) \cap R(q) \rvert$
RR	somme de l'inverse des rangs des documents dans $M(e) \cap R(q)$
BordaFuse	somme des K-rangs des documents dans $M(e) \cap R(q)$
CombMED	médiane des scores des documents dans $M(e) \cap R(q)$
CombMIN	minimum des scores des documents dans $M(e) \cap R(q)$
CombMax	maximum des scores des documents dans $M(e) \cap R(q)$
CombSUM	somme des scores des documents dans $M(e) \cap R(q)$
CombANZ	$CombSUM / M(e) \cap R(q)$
CombMNZ	$\lvert M(e) \cap R(q) \rvert$. CombSUM
expCombSUM	somme des exp des scores des documents dans $\lvert M(e) \cap R(q) \rvert$
expCombANZ	$expCombSUM / \lvert M(e) \cap R(q) \rvert$
expCombMNZ	$\lvert M(e) \cap R(q) \rvert \cdot$ expCombSUM

TABLE 2.2 – Résumé des techniques de fusion de données et des modèles de vote

2.4.2.3 Autres modèles

D'autres modèles de RI ont été appliqués à la recherche d'experts, tels que la modélisation des sujets (topic modeling)[95], le modèle d'espace vectoriel (vector space)[23], et la recherche par clusters (cluster-based retrieval)[96].

Quant à la modélisation des sujets, de nombreux modèles ont été proposés. L'une des premières tentatives a été le modèle auteur-sujet (ATM). Dans cette approche, les auteurs et le contenu sont modélisés simultanément avec des hyperparamètres couplés pour l'expertise des auteurs et les thèmes présents dans le texte. Après l'ATM, le modèle auteur-personnatopique (APT) est apparu, qui est une version étendue de l'APT, en ajoutant une couche supplémentaire de variables non observées appelées "personas". Dans une extension du modèle ATM, des modèles auteur-conférence-sujet (ACT) ont été proposés, ce modèle incorporant la variable observée "conférence". Un modèle de mélange basé sur l'analyse sémantique latente probabiliste (PLSA) a également été proposé. En outre, le modèle citation-auteur-sujet (CAT) a été proposé. Ce modèle modélise les informations sur l'auteur cité ainsi que les mots et les auteurs.

Quant au modèle d'espace vectoriel (VSM), les experts sont représentés comme une combinaison linéaire de documents, ce qui donne des vecteurs experts qui se trouvent dans le même espace vectoriel que les documents et les requêtes.

Et enfin un modèle de recherche basé sur les clusters. Il est très similaire aux modèles de documents mais dans un contexte différent.

Un autre type de modèle a été proposé, celui basé sur les graphes. Les modèles basés sur le graphe déterminent les associations entre les requête et les personnes par inférence sur un

graphe d'expertise, comprenant des documents, des candidats experts et différentes relations.

2.4.3 Les bases de données utilisées

Dans cette section nous présentons certaines des bases de données qui ont été utilisées pour évaluer les systèmes de RE. La disponibilité des données est un des facteurs les plus importants pour le développement d'une tâche de recherche d'informations. Une partie de ces BD est apparue après la TREC, certaines d'entre elles ont été développées par des chercheurs indépendants. Parmis ces BD nous citons la DBLP [5] combinee avec Google scholar [97], DBLP en RDF [6] [98], Yahoo Answe.or [99],ArnetMiner [100]. Les auteurs de la reference [5] ont développé leur propre base de donnees en combinant les informations depuis Arxive [101] et MAG [7].
Le contenu des bases de données citées est présenté dans le tableau suivant[102] :

BD	# Experts	# Documents	# Requête	# Jugements de pertinence	Taille	Publique
W3C [8]	1092	331037	99	9860	5.7	Oui
DBLP in RDF	17910795	715690	-	-	19	No
INDURE	12535	-	100	6482	-	No
DBLP+G scholar	574.369	953774	17	244	20	Non
Arnet Miner	1033050	1632440	13	1781	0.85	Oui
Yahoo ! Answers	169819	780193	67	7515	-	Oui

TABLE 2.3 – Aperçu des bases de données

2.5 Conclusion

Dans ce chapitre, nous avons présenté quelques techniques et modèles utilisés dans la recherche d'information, ainsi que les métriques utilisées pour les évaluations. Ensuite, nous avons montré comment les réseaux de neurones ont contribué au traitement automatique du langage et à la recherche d'information. Enfin, la dernière partie de ce chapitre décrivait le domaine de la recherche d'experts. Dans le prochain chapitre, nous allons présenter en détail des approches que nous proposons pour notre système de RE.

5. DataBase systems and Logic Programming
6. Resource Description Framework
7. Microsoft Academic Graph

Chapitre 3

Approches proposées pour la recherche d'experts académiques

3.1 Introduction

A travers ce chapitre nous allons présenter les trois principales parties où s'est située notre contribution, chacune d'entre elles touche à un niveau de la recherche. En commençant par l'étape de l'indexation par phrases. Ensuite, l'étape de modification à la formule de vote afin de limiter l'impact des auteurs très prolifiques, et enfin au niveau de la requête introduite, à laquelle nous ajoutons sa définition dans le processus de recherche, de deux manières différentes.

3.2 Représentation des documents et calcul de similarité

Plusieurs méthodes ont été utilisées pour représenter les documents d'un auteur, nous les divisons en deux catégories : une approche de base, et notre approche proposée. Chacune de ces approche utilise le plongement d'une manière spécifique, aboutissant à différentes manières de calculer la similarité entre une requête introduite par l'utilisateur et le document (sim_{QD}), cette similarité sera utilisée dans la formule de vote que nous présenterons un peu plus loin.

3.2.1 Approche de base

Le modèle de base utilisé pour les comparaisons est issu des travaux de Berger et al.[5]. Nous avons utilisé leur meilleur résultat, appelé par ses auteurs *separate embedding strategy*, qui est de représenter un document par un vecteur v_{doc} en prenant la moyenne du plongement de son résumé, avec le plongement de son titre. En d'autres termes, le plongement du résumé $v_{résume}$ et du titre v_{titre} sont calculés séparément, pour ensuite faire leur somme et diviser par 2.

$$v_{doc} = (v_{resume} + v_{titre})/2$$

Dans cette approche, la similarité entre requête et document (sim_{QD}), est la même que la similarité entre le le vecteur représentatif du document v_{doc} avec le vecteur représentatif de la requête v_q :

$$sim_{QD} = sim(v_{doc}, v_q)$$

3.2.2 Indexation par phrases

Un document est composé de plusieurs phrases, et chacune d'entre elles a un apport différent dans l'idée globale du paragraphe. L'idée derrière notre méthode est de limiter l'impact des phrases non significatives par rapport à la requête, et voir si ça peut limiter la perte d'information causée par la moyenne utilisée dans de l'approche de base.

Après avoir séparé un document en ses phrases, nous appliquons l'algorithme suivant sur chacune d'entre elles :

Algorithm 1 Plongement et normalisation

Require: ph : une chaîne de caractères
Ensure: v : vecteur représentatif normalisé (L2)
choisir un modèle de plongement SciBert ou RoBerta
Encoder ph en utilisant m afin d'obtenir son vecteur v représentatif.
Appliquer la normalisation L2 sur v

Le résultat sera un ensemble de vecteurs normalisés qui concernent le document. Ces vecteurs seront par la suite ajoutés à un index de la librairie FAISS. Nous répétons ce processus pour tous les documents de la base de données, bien sûr l'information d'appartenance de chaque phrase à quel document est aussi sauvegardée.

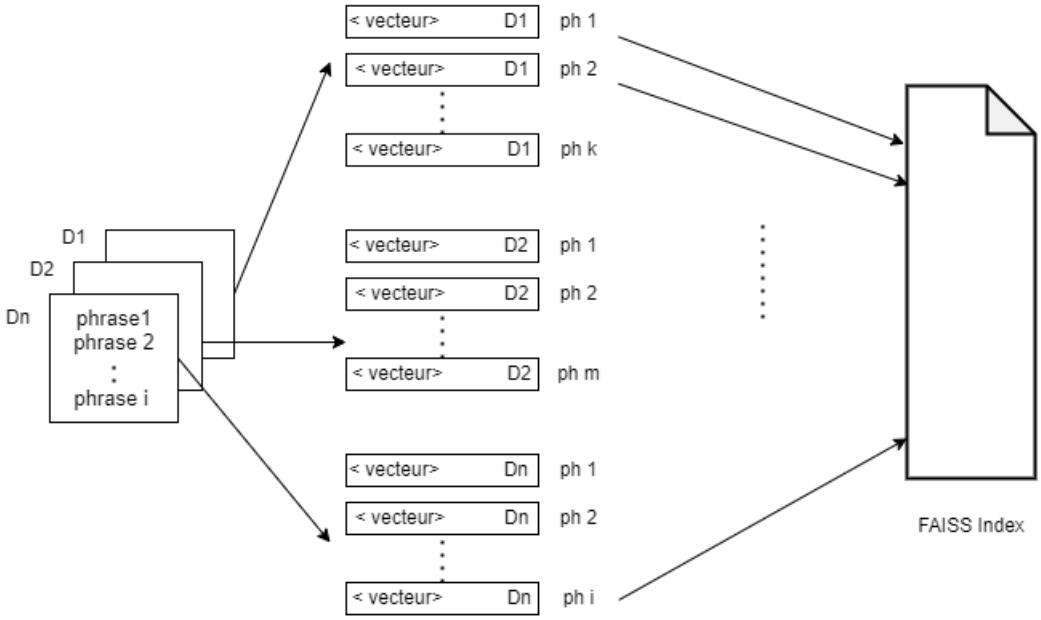


FIGURE 3.1 – Schéma de l'indexation par phrases

Après l'introduction d'une requête par l'utilisateur, le résultat de la recherche sera les phrases les plus proches de la requête. La prochaine étape est d'affecter un score pour chaque document sélectionné à partir de ses phrases retournées, nous avons essayé trois méthodes (les détails de leurs efficacités seront présentés dans le chapitre suivant) :

— moyenne : le score du document est la moyenne des scores de ses phrases retournées.

$$sim_{QD} = moyenne(sim(v_q, v_{phi}))$$

ou :
v_{phi} : est le vecteur de la phrase i, retournée par la recherche sur l'index.
$sim(v_q, v_{phi})$: la similarité entre la requête et la phrase i.

— max :le score du document est le score max de ses phrases retournées.

$$sim_{QD} = max(sim(v_q, v_{phi}))$$

ou :
v_{phi} : est le vecteur de la phrase i, retournée par la recherche sur l'index.
$sim(v_q, v_{phi})$: la similarité entre la requête et la phrase i du document

— somme : le score du document est la somme des scores de ses phrases retournées.

$$sim_{QD} = somme(sim(v_q, v_{phi}))$$

ou :
v_{phi} : est le vecteur de la phrase i, retournée par la recherche sur l'index.
$sim(v_q, v_{phi})$: la similarité entre la requête et la phrase i du document

Le schéma suivant résume l'étape d'affectation du score, avec par exemple la stratégie max :

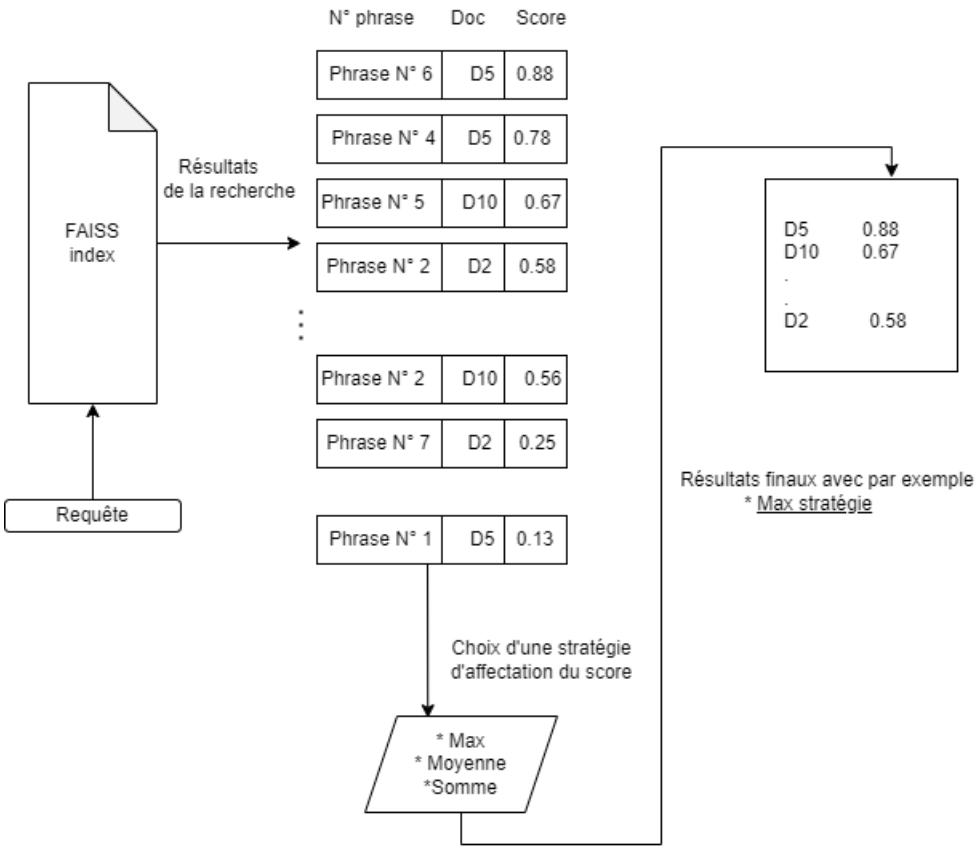


FIGURE 3.2 – Schéma de l'affectation du score à un document à partir de ses phrases retournées (exemple : stratégie max)

On a aussi appliqué une normalisation qui consiste à diviser le score attribué au document, par le nombre de ses phrases.

3.2.2.1 Recherche et stockage avec FAISS

Pour stocker tous les vecteurs que nous avons obtenus, ainsi que pour la recherche de similarité avec la requête, nous avons utilisé la bibliothèque d'indexation FAISS de FACEBOOK[103].

Cette bibliothèque est optimisée pour la recherche dans des espaces de larges dimensions, et se base sur la distance euclidienne [1]. Plus spécifiquement, nous avons customisé la classe IndexFlatL2, pour la rendre compatible avec l'indexation par phrases, en y ajoutant néanmoins, les outils qui permettent d'identifier les documents après l'obtention du résultat.

La requête introduite par l'utilisateur est sous format textuelle, nous appliquons l'algorithme plongement et normalisation précédent, pour la transformer en un vecteur normalisé v_q. Nous utiliserons l'index FAISS qui contient les phrases de tous les documents du corpus, pour trouver les vecteurs les plus proches de la requête. La normalisation L2 garantie l'obtention de distance euclidienne comprises entre 0 et 2 :

$$distance(v_q, v_{ph}) \in [0,2]$$

ou
v_q : vecteur normalisé de la requête.
v_{ph} : vecteur normalisé d'une phrase indexée avec FAISS.

Après avoir obtenu les distances entre v_q et v_{ph}, nous les transformons en des score de similarité par la formule suivante :

$$sim(v_q, v_{ph}) = 1 - distance(v_q, v_{ph})/2$$

Le score de similarité sera compris entre 0 et 1 :

$$sim(v_q, v_{ph}) \in [0,1]$$

3.2.2.2 formule de vote pour calcul du score pour l'expert

Après avoir déterminé les documents les plus similaires à une requête, nous utilisons un modèle de vote pour affecter un score final pour chaque auteur à partir de ces documents. La formule que nous avons utilisé est celle de la somme des exponentielle des scores des documents (expoCombSum), qui est considérée avec expCombMNZ, comme l'une des formules les plus performantes parmi celles existantes et présentées la TABLE 2.2 du chapitre précédent [92] :

$$score_{AQ} = \sum_{i=1}^{N} e^{sim_{QD_i}}$$

ou
$score_{AQ}$: score attribué à l'auteur A vis-a-vis de la requête Q.
sim_{QD_i} : score attribue au document D_i vis-a-vis de la requête Q.
N : nombre de documents.

Macdonald et Ounis [104] ont supposé que chaque document produit un score statique à chacun de ses auteurs, en d'autre termes, que chaque auteur a eu la même contribution dans la publication. Or, il pourrait être utile d'attribuer des pondérations différentes aux différents auteurs d'un même document. Berger et al. [5], ont utilisé l'ordre dans lequel les noms des auteurs apparaissent pour définir, le taux de contribution de chacun. Néanmoins, des recherches récentes ont montré qu'en raison de l'interdisciplinarité croissante de la recherche, il est de plus en plus difficile d'évaluer les auteurs en se basant sur ce genre de critère [105]. Nous proposons dans ce qui suit une nouvelle méthode de distribution du score, suivie d'une modification de la formule de vote précédente.

1. En géométrie des coordonnées, la distance euclidienne est la distance entre deux points.

3.2.3 Domaine de l'auteur pour la distribution du score

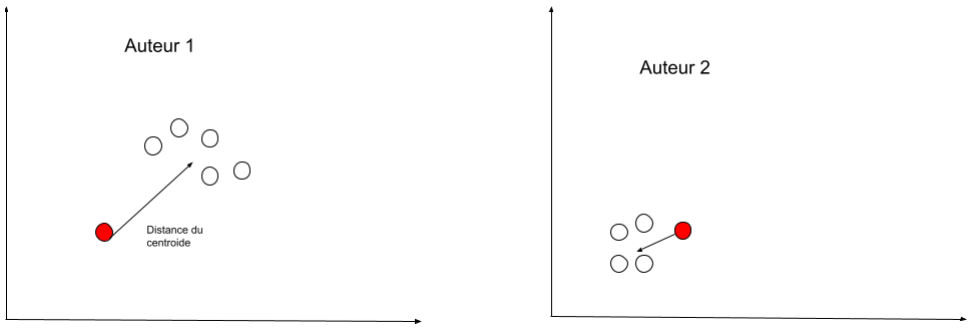
Notre méthode a pour but de quantifier l'apport de chaque auteur dans un article, nous nous basons sur l'hypothèse que dans le cas où plusieurs auteurs apparaissent dans un document D, celui avec le plus de publications proches du sujet de D, est considéré comme ayant le plus d'expertise et donc recevra une plus grosse part de sim_{QD}. Notre méthode a aussi comme effet de limiter l'influence des auteurs se trouvant dans plusieurs publications. Pour ce faire, nous utiliserons les publications de chaque auteurs (le résumé et le titre), pour créer un profile :

$$Profile(p) : \{v_{doc_i}, 0 < i < N\}$$

tel que : N est le nombre de publications de p
v_{doci} est le vecteur représentatif du document i, obtenu comme suit :

$$v_{doc_i} = (v_{résume} + v_{titre})/2$$

ou, $v_{résume}$ et v_{titre} sont respectivement les plongement normalisés du résumé et du titre, obtenus par l'algorithme 1.
Étant donné qu'on utilise le même modèle de plongement (scibert) pour tous les profils, les v_{doci} appartiennent au même espace vectoriel de 768 dimensions. Pour illustrer notre méthode, nous allons nous contenter de seulement deux. Le graphe suivant représente le profil (cluster) de l'auteur, chaque point est le vecteur représentatif d'un des documents dans lesquels cet auteur apparaît, les flèches sont les distances entre un point donné et le centroïde du cluster :



Le point rouge représente un document auquel nous venons de calculer le score, dans ce document le nom de auteur 1 et 2 y figurent. Nous pouvons remarquer facilement que le vecteur représentatif (le point rouge) est plus éloigné du centroïde du groupe de l'auteur 1 que celui de l'auteur 2. Ainsi lors de la distribution du score nous réserverons une meilleure part à l'auteur 2 qu'à son collègue. Ceci se concrétisera en la modification de la formule de vote vue dans la partie précédente, en introduisant une nouvelle grandeur qui est la distance entre auteur et document $dist_{ADi}$:

$$score_{AQ} = \sum_{i=1}^{N} e^{sim_{QD_i}/dist_{AD_i}}$$

ou $dist_{ADi}$ est la distance euclidienne entre le centroïde de l'auteur et le document i, a laquelle one ajoute 1 pour eviter la devision par 0.

$$dist_{AD_i} = dist(centroide, D_i) + 1$$

3.2.4 Techniques proposées pour l'expansion de la requête

Pour améliorer notre système, nous avons introduit le concept d'expansion des requêtes. Nous avons utilisé Wikipédia comme source d'expansion. Lorsque l'utilisateur saisit une requête

(un concept), le système obtient sa définition (résumé) de cette requête à partir de Wikipedia, en utilisant l'API Wikipedia.

Wikipedia-API : Est un wrapper Python facile à utiliser pour l'API de Wikipédia. Il peut extraire du texte, des sections, des liens, des catégories, des traductions et d'autres informations de Wikipédia. Pour les situations d'utilisation les plus typiques, la documentation inclut des extraits de code[106].

Pour l'expansion de la requête, nous avons mis en œuvre deux techniques, l'une appelée méthode "hybride", et l'autre, méthode "moyenne". Dans les sections qui suivent, nous exposons en détail les deux approches, avec des schémas et des algorithmes détaillés.

3.2.4.1 Approche proposée : expansion "hybride" de la requête

Nous récupérons les k meilleures phrases (phrases pour l'indexation par phrases, et documents pour l'indexation par documents) liées au concept et les k meilleurs phrases liés à sa définition, nous les fusionnons et à la fin nous prenons les k meilleurs selon le score. le score est calculé avec le forum suivant :

$$Score = \alpha.s_1 + \beta.s_2$$

Où :
s_1 : est le score de la phrase (phrase pour l'indexation par phrases, et document pour l'indexation par documents) dans la liste des scores du concept.
s_2 : est le score de la (phrase pour l'indexation par phrases, et document pour l'indexation par documents) dans la liste des scores de sa définition.
α et β : $0 < \alpha < 1$ et $0 < \beta < 1$, tel que $\alpha + \beta = 1$.
Si la phrase (phrase pour l'indexation par phrases, et documente pour l'indexation par documentes) n'appartient pas à l'une des listes, alors il prend le score minimum de cette liste.

L'algorithme "Get relevant sentences using the hybride method", et le diagramme de la figure 3.2, détaillent le principe de cette approche.

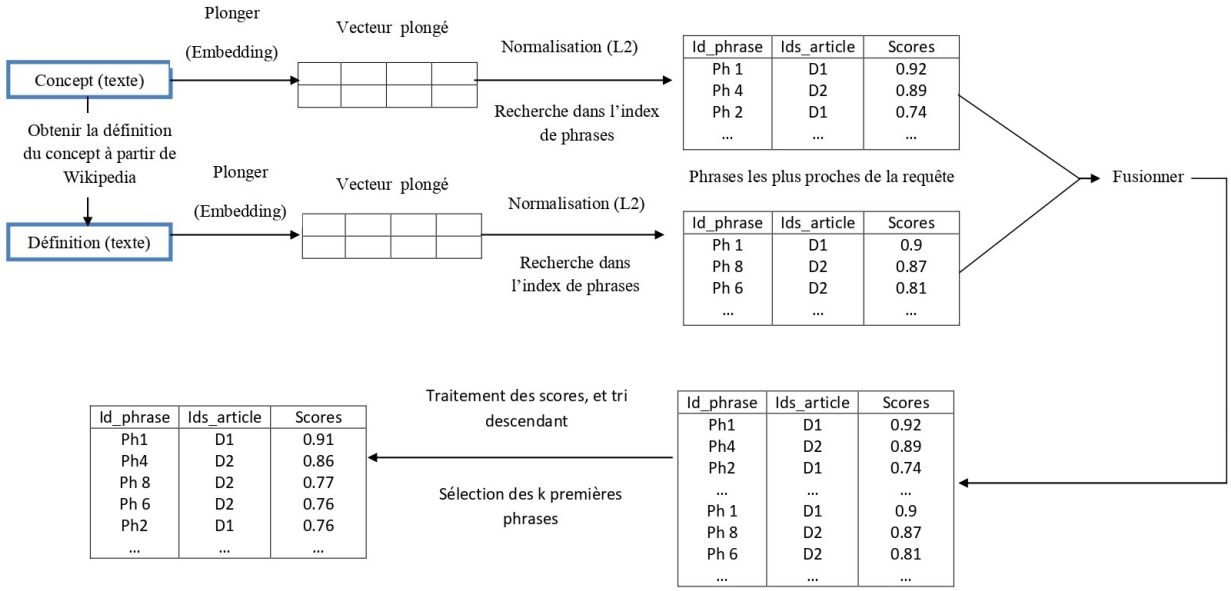


FIGURE 3.3 – Schéma représentatif de la méthode "hybride"

Algorithm 2 Get relevant sentences using the hybride method

Require: Concept, definition, k, α, β
Ensure: A dataframe of relevant sentences, with the columns : idPaper, idSentence, score.

$ConceptEmbedding = Embedding(concept)$
$ConceptNormalized = Normalize(ConceptEmbedding)$
$ConceptIdsAndScores = SearchInIndex(ConceptNormalized, k)$
$DefinitionEmbedding = Embedding(definition)$
$DefinitionNormalized = Normalize(DefinitionEmbedding)$
$DefinitionIdsAndScores = SearchInIndex(DefinitionNormalized, k)$
$minConcept = minimumScore(ConceptIdsAndScores)$
$maxConcept = maximumScore(DefinitionIdsAndScores)$
$IdsAndScores = Concat(ConceptIdsAndScores, DefinitionIdsAndScores)$
$IdsAndScores = RemoveDuplicates(IdsAndScores)$
for each $phraseIdinIdsAndScores$ **do**
    if $phraseIdexistsinConceptIdsAndScores$ **then**
        $s_1 = ConceptIdsAndScores[score]$
    else
        $s_1 = minConcept$
    end if
    if $phraseIdexistsinDefinitionIdsAndScores$ **then**
        $s_2 = DefinitionIdsAndScores[score]$
    else
        $s_2 = minConcept$
    end if
    $s = \alpha * s_1 + \beta * s_2$
    $NewIdsAndScores.add([phraseId, s])$
end for
$FinalIdsAndScores = GetTopK(NewIdsAndScores, k)$

3.2.4.2 Approche proposée : expansion ”moyenne” de la requête

Cette contribution est basée sur le fait que nous prenons la moyenne des deux vecteurs plongés (embedded vectors) : vecteur de la requête initiale et vecteur de sa définition.
L'algorithme ”Get relevant sentences using the hybrid methodGet relevant sentences using the mean method”, et le diagramme de la figure 3.3, détaillent le principe de cette approche.

Algorithm 3 Get relevant sentences using the mean method

Require: Concept, definition, k
Ensure: A dataframe of relevant sentences, with the columns : idPaper, idSentence, score.

$$ConceptEmbedding = Embedding(concept)$$
$$DefinitionEmbedding = Embedding(definition)$$
$$VectorEmbedded = (ConceptEmbedding + DefinitionEmbedding)/2$$
$$VectorNormalized = Normalize(VectorEmbedded)$$
$$IdsAndScores = SearchInIndex(VectorEmbedded, k)$$

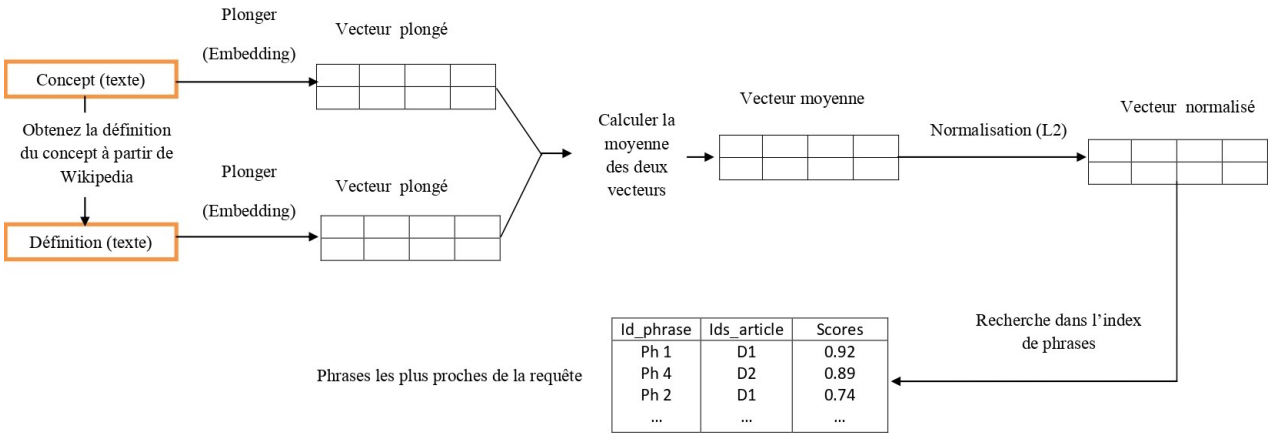


FIGURE 3.4 – Schéma représentatif de la méthode ”moyenne”

3.3 Conclusion

Dans ce chapitre nous avons détaillé les trois principales contributions apportées dans le cadre de notre PFE, qui pour rappel sont l'indexation par phrases avec les différentes stratégies d'attribution du score utilisées : ”Moyenne”, ”Maximum”, ”Somme”, dans le chapitre suivant nous évaluerons l'efficacité de chacune d'entre elle et comparons avec l'approche de base, et ce en utilisant deux bases de données différentes. L'autre contribution concerne l'expansion de la requête, avec les deux méthodes proposées qui sont l'expansion hybride ainsi que l'expansion moyenne de la requête les évaluations de ces deux méthodes et une comparaison avec le cas sans expansion et ce avec tous les systèmes utilisés sera présenté aussi dans le chapitre suivant. Nous avons aussi lors de ce chapitre proposé une distribution du score d'un document sur ses auteurs, néanmoins pour utiliser cette fonctionnalité nous avons eu besoin d'un nombre représentatif de documents pour chaque auteur afin de capturer au mieux son expertise. Ainsi, dans le chapitre suivant nous présenterons les détails techniques d'obtention de notre nouvelle base de données ACM ainsi que plus de détails sur les limitations de l'autre base utilisée lors des évaluations qui est Arxiv+MAG.

Chapitre 4

Réalisation et évaluation des approches proposées

4.1 Introduction

Dans ce chapitre, nous allons au départ présenter les différents outils utilisés dans notre projet.
Ensuite, Ensuite, nous allons présenter les résultats des évaluations de chacune des deux bases de données utilisées (Arxive+MAG, ACM) avec notre système de recherche d'expert, qui utilise l'indexation par phrase, et le système des auteurs [5] qui utilise l'indexation par document, en introduisant pour chaque système d'autres contributions, à savoir les techniques d'expansion de la requête proposées. En d'autres termes, nous allons comparer les deux types d'indexation, et pour chaque type, nous expérimentons, l'apport des différents types d'expansion de requête : Moyenne, Hybride (plus de détails dans le chapitre précédent).
La structure suivie par la partie évaluation est comme suit : une section pour la base de donnée, suivi par une sous-section pour le modèle de plongement (*RoBERTa*, *SciBERT*), où nous présentons les résultats de l'indexation par phrases et par document, d'abord séparément, ensuite comparer et interpréter les meilleurs résultats des deux types d'indexation, en dernier, nous verrons si *SciBERT* (entraîné sur des documents scientifiques) est meilleur que *RoBERTa* avec nos deux bases de données.
Enfin, nous parlerons de la robustesse de notre système, au changement des bases de données, ainsi qu'aux futurs travaux à apporter.
Le code source des approches est disponible sur : `https://github.com/chakibMH/PFE_CODE`.

4.2 Environnement de travail

Dans cette section, nous présentons les environnements de travails matériels et logiciels sous lesquels nous avons effectuées nos réalisations et nos évaluations.

4.2.1 Environnement matériel

L'implémentation et l'évaluation de notre approche étaient réalisées sur deux machines, avec la configuration suivante :

Machine 01 :
— Processeur : RYZEN 7 4800 h, 4 cœurs, 16 threads
— Mémoire RAM : 8.00 Go

— Carte graphique : Nvidia 1660ti, mémoire du processeur graphique 10 Go
— Système d'exploitation : Windows 10

Machine 02 :

— Processeur : Intel(R) Core (TM) i7-7500U CPU @2.70GHz, 2 cœurs, 4 threads
— Mémoire RAM : 8.00 Go
— Carte graphique : Radeon (TM) 530, mémoire du processeur graphique 6.0 Go
— Système d'exploitation : Windows 10

4.2.2 Environnement logiciel

Python est un langage de programmation, qui a été créé par Guido van Rossum, et publié en 1991. Il peut être utilisé sur un serveur pour créer des applications web, à côté de logiciels pour créer des flux de travail, pour se connecter à des systèmes de bases de données. Il peut également lire et modifier des fichiers, traiter des données volumineuses et effectuer des calculs mathématiques complexes. Il peut aussi être mis à profit pour le prototypage rapide ou le développement de logiciels prêts à être mis en production. Python est si populaire pour les raisons suivantes : il fonctionne sur différentes plateformes (Windows, Mac, Linux, Raspberry Pi, etc.), sa syntaxe est simple et similaire à celle de la langue anglaise, sa syntaxe permet aux développeurs d'écrire des programmes avec moins de lignes que certains autres langages de programmation, et il fonctionne sur un système d'interprétation, ce qui signifie que le code peut être exécuté dès qu'il est écrit. Cela signifie que le prototypage peut être très rapide. Python peut être traité de manière procédurale, orienté objet ou fonctionnelle[107].

Anaconda est une plateforme de distribution Python open source. Elle permet de rechercher et d'installer facilement plus de 8 000 paquets Python/R et d'accéder à une vaste bibliothèque de contenu communautaire et de support dans le domaine de la science des données et de l'apprentissage automatique. Anaconda est construit et compilé pour tous les principaux systèmes d'exploitation et architectures[108].

Spyder est un environnement scientifique gratuit et open source écrit en Python, pour Python, et conçu par et pour les scientifiques, les ingénieurs et les analystes de données. Il offre une combinaison unique des fonctionnalités avancées d'édition, d'analyse, de débogage et de profilage d'un outil de développement complet avec l'exploration de données, l'exécution interactive, l'inspection approfondie et les magnifiques capacités de visualisation d'un progiciel scientifique[109].

CUDA (*Compute Unified Device Architecture*) est une plate-forme de calcul parallèle et un modèle de programmation développés par NVIDIA pour le calcul général sur les processeurs graphiques (GPU). CUDA permet d'accélérer considérablement les applications informatiques en exploitant la puissance des GPU. Elle accélère les applications dans un large éventail de domaines, du traitement d'images à l'apprentissage profond, en passant par l'analyse numérique et les sciences informatiques. Dans les applications accélérées par les GPU, la partie séquentielle de la charge de travail s'exécute sur le CPU, tandis que la partie intensive en calcul de l'application s'exécute sur des milliers de cœurs GPU en parallèle. CUDA peut être utilisé avec de nombreux langages tels que C, C++, Fortran, Python et MATLAB et exprimer le parallélisme par le biais d'extensions sous la forme de quelques mots-clés de base. La boîte à outils CUDA comprend des bibliothèques accélérées par le GPU, un compilateur, des outils de développement et

le runtime CUDA.[110].

FAISS (*Facebook AI Similarity Search*) est une bibliothèque qui permet de rechercher rapidement les incorporations de documents multimédia qui sont similaires les unes aux autres. Elle résout les limitations des moteurs de recherche traditionnels qui sont optimisés pour les recherches basées sur le hachage, et fournit des fonctions de recherche de similarité plus évolutives. Il comprend des implémentations de la recherche du plus proche voisin pour des ensembles de données de plusieurs millions à plusieurs milliards d'unités, qui optimisent le compromis entre la vitesse de la mémoire et la précision. FAISS contient des algorithmes de recherche dans des ensembles de vecteurs de n'importe quelle taille, ainsi que du code de soutien pour l'évaluation et le réglage des paramètres. Certains de ses algorithmes les plus utiles sont implémentés sur le GPU. FAISS est implémenté en C++, avec une interface Python optionnelle et un support GPU via CUDA[103].

Sentence Transformers est un framework fournissant une méthode facile pour calculer des représentations vectorielles pour les phrases, les paragraphes et les images. Les modèles sont basés sur des réseaux de transformateurs comme *BERT* / *RoBERTa* / *XLM-RoBERTa* etc. et atteignent des performances de pointe dans diverses tâches. Le texte est intégré dans l'espace vectoriel de sorte que les textes similaires sont proches et peuvent être trouvés efficacement en utilisant la similarité du cosinus. Il fournit un nombre croissant de modèles pré-entraînés à la pointe de la technologie pour plus de 100 langues[111].

Scikit-learn est une bibliothèque open source d'apprentissage automatique, construite sur NumPy, SciPy et matplotlib. Elle prend en charge l'apprentissage supervisé et non supervisé. Elle fournit également divers outils pour l'ajustement des modèles, le prétraitement des données, la sélection des modèles, l'évaluation des modèles et de nombreux autres utilitaires[112].

Requests est une bibliothèque HTTP pour le langage de programmation Python. Elle permet d'envoyer des requêtes HTTP/1.1 extrêmement facilement. Il n'est pas nécessaire d'ajouter manuellement des chaînes de requête aux URL, ni de coder les données PUT et POST - de nos jours, il suffit d'utiliser la méthode json[113].

.

Beautiful Soup est une bibliothèque Python permettant d'extraire des données de fichiers HTML et XML. Elle fonctionne avec un analyseur syntaxique pour fournir des méthodes idiomatiques de navigation, de recherche et de modification de l'arbre d'analyse.

Wikipedia-API est un wrapper Python facile à utiliser pour l'API de Wikipédia. Il permet d'extraire des textes, des sections, des liens, des catégories, des traductions, des résumés, etc. de Wikipédia[114].

.

.

Tant d'autres bibliothèques python ont été employées telles que : **RegEx**, **Pandas**, **Numpy**, **num2words**, **math** ... ect.

4.3 Optimisations

4.3.1 Multiprocessing

Comme le *web-scraping*, le nettoyage des données et l'évaluation sont des tâches qui ont un temps d'exécution remarquable et pour remédier à ceci, nous nous sommes servis du Multiprocessing qui est l'utilisation simultanée de plusieurs processus afin de résoudre certains problèmes en un temps raisonnable.
Le graphique suivant montre la différence entre l'utilisation d'un seul processus et quatre processus dans le web scraping :

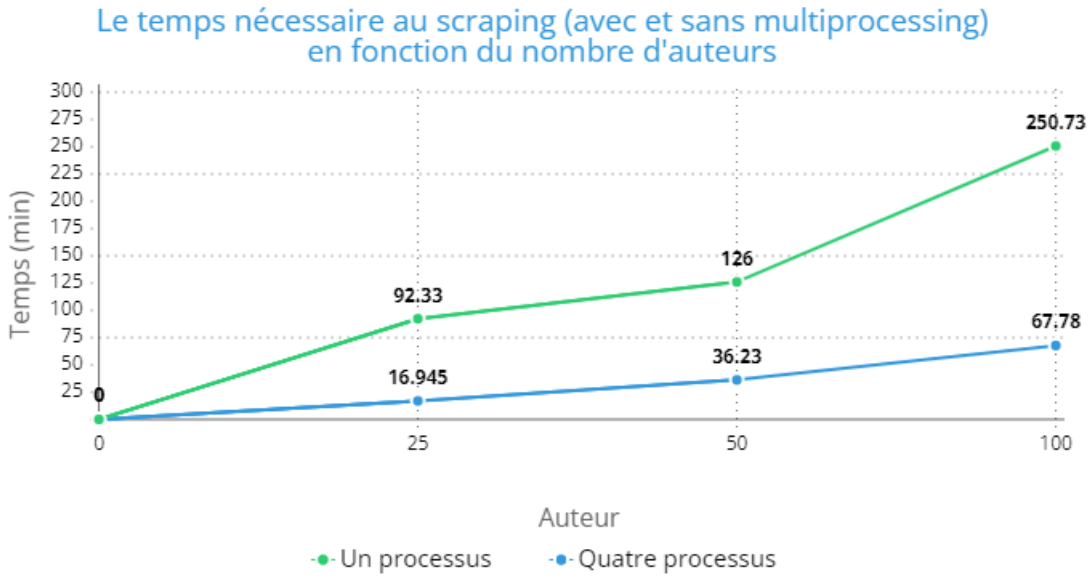


FIGURE 4.1 – Un graphique qui représente le temps nécessaire au scraping (avec et sans multiprocessing) en fonction du nombre d'auteurs.

4.3.2 CUDA (Compute Unified Device Architecture)

CUDA est une technologie GPGPU [1], c'est-à-dire l'utilisation d'un GPU pour effectuer des calculs généraux à la place du CPU. Cette technologie nous a donné l'avantage de pouvoir effectuer des calculs des plongements très rapidement, ce qui nous a permis d'économiser de très nombreuses journées d'exécution. Le graphique ci-dessous(figure 4.2), nous montre qu'il y a un grand écart entre le temps d'exécution sur un CPU et un GPU.

1. *General-Purpose Computing on Graphics Processing Units*

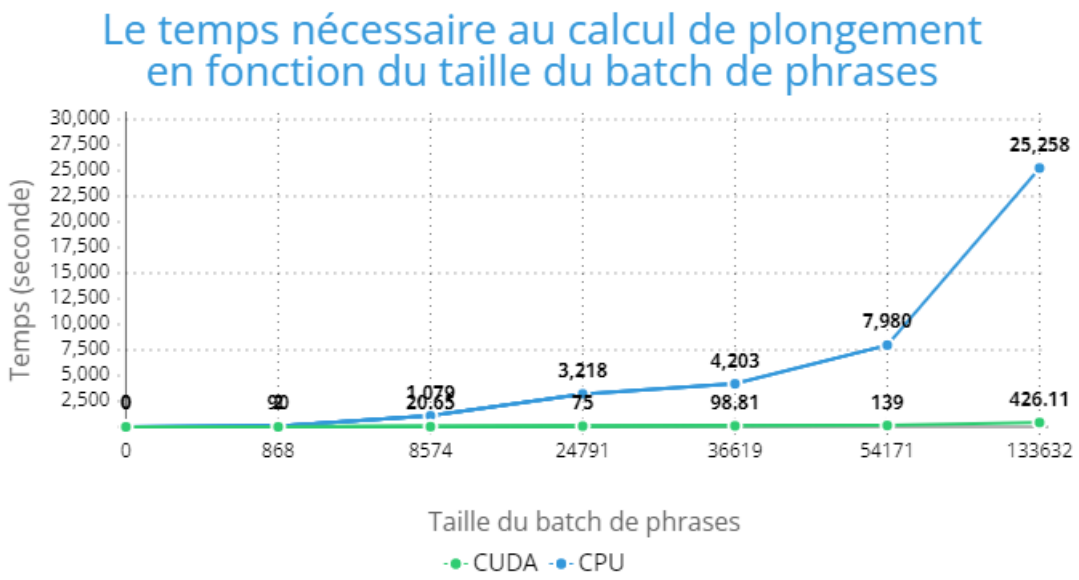


FIGURE 4.2 – Un graphique qui représente le temps nécessaire au calcul de plongement en fonction du taille du batch de phrases.

4.4 Datasets utilisées

Dans cette expérience, nous avons eu recours à deux collections de données, l'une provenant d'ACM et l'autre d'arXiv [2]+MAG [3].
Concernant notre travail, nous nous sommes limité dans le domaine de l'informatique, car c'est le domaine dont nous sommes très proches et nous pouvons mieux interpréter, une autre raison, c'est d'extraire les données du site web ACM Digital Library, qui se spécialise sur les publications liées à l'informatique, et nécessite pas une clé d'accès spécifique, par rapport aux d'autres sites web comme Scopus, ou IEEE. Aussi, la base de données Arxiv+MAG, était déjà en libre d'accès, lors du début de nos investigations, et elle était consacrée exclusivement au domaine informatique.
Vous trouverez, ci-après, un récapitulatif sur les deux bases de données utilisées.

4.4.1 Dataset Arxiv + MAG

La base de données Arxiv+MAG est une collection déjà organisée et nettoyée par berger et al. [5] de la manière suivante :
Ils ont extrait les articles relatifs à l'informatique en effectuant une correspondance exhaustive des titres sur l'ensemble des données MAG avec les 113 864 titres d'articles obtenus d'Arxiv. Cette recherche a permis d'obtenir 29 237 articles. Ensuite, ils ont étendu cet ensemble avec les références de tous les articles, ce qui a donné un ensemble de 221 347 articles. À partir de ces 221 347 articles, ils ont ensuite effectué un échantillonnage stratifié limité pour les auteurs afin d'extraire un sous-ensemble de 5 000 auteurs représentatifs des populations d'auteurs très,

2. **arXiv** est archivée en libre accès de 2 113 816 articles scientifiques dans plusieurs domaines. Il offre aux chercheurs un large éventail de services : soumission, compilation, production, extraction, recherche et découverte d'articles, distribution sur le web pour les lecteurs humains et accès API pour les machines, ainsi que conservation et préservation du contenu[115].Le contenu de ce site n'a pas été examiné par les pairs par Arxiv (*peer-reviewed*)[116]

3. Microsoft Academic Graph est un graphe hétérogène contenant des publications scientifiques, des relations de citation entre ces publications, ainsi que des auteurs, des institutions, des revues, des conférences et des domaines d'étude. Ce graphique est utilisé pour alimenter les fonctionnalités de Bing, Cortana, Word et Microsoft Academic[117]

moyennement et moins prolifiques. Pour ces auteurs, ils ont récupéré tous leurs articles et leurs références, ce qui a donné un ensemble de 127 716 articles, incluant les auteurs des articles référencés. Pour tous ces nouveaux auteurs, ils ont collecté les métadonnées de l'ensemble de données des auteurs MAG et ont agrégé ces informations dans un seul ensemble de données des auteurs finaux (67808 auteurs).

4.4.2 Dataset ACM

Pour mieux tester notre approche, nous avons collecté un ensemble de données à partir de la bibliothèque numérique de l'ACM.

Notre base de données contient tous les résumés des publications disponibles pour un auteur, ainsi que les revues, les actes de conférence, les magazines techniques, les bulletins d'information et les livres, portant exclusivement sur le domaine de l'informatique. Elle dispose d'un ensemble très riche de connexions entre auteurs, œuvres, institutions et communautés spécialisées[118]. De plus, elle définit pour chaque auteur son domaine d'expertise par l'utilisation de tags et de mots clés très précis.

Nous avons sélectionné un ensemble aléatoire de 13 931 auteurs dans le domaine de l'informatique, appartenant à la base de données d'ACM. La raison de la sélection de ce large ensemble d'auteurs est qu'un plus grand nombre d'articles est bénéfique pour la recherche en raison de l'espace de recherche plus grand (plus de détails dans la section 4.7).

4.5 Méthodes d'évaluation

L'ensemble des requêtes entrant dans l'évaluation, proviennent des tags des auteurs (domaines d'activités et mots-clés). La raison de ce choix est l'absence de dataset qui relie chaque requête manuellement introduites dans un ensemble bien défini d'experts ciblés, manuellement défini par d'autres experts humains.
Deux méthodes de contrôle de la pertinence ont été appliquées.

4.5.1 Évaluation par des requêtes thématiques exactes

Ce protocole d'évaluation consiste à utiliser les noms ou les courtes descriptions de les sujets associés à un expert comme requêtes brutes afin de classer l'ensemble des candidats disponibles. La requête sera comparée aux tags d'un auteur, et si une correspondance exacte existe, alors cet auteur est jugé pertinent. Dans la partie résultats, cette méthode sera référencée comme ***Exact***.

4.5.2 Évaluation avec approximation de la requête

Cette méthode ne correspond pas exactement à la requête à l'ensemble des tags d'un auteur. En revanche, elle utilise une méthode de vérification de la pertinence floue selon laquelle, à partir d'une requête, nous devons calculer la similarité en cosinus [4] entre le plongement de la requête et chacun des tags de l'auteur. Si l'une des similarités est supérieure à un seuil choisi, l'auteur est jugé pertinent. Dans la partie résultats, cette méthode sera référencée comme ***Approximate***.

4. La similarité cosinus est une métrique qui modélise un texte comme un vecteur de termes, et la similarité entre deux textes est dérivée de la valeur du cosinus entre les vecteurs de termes de deux textes[119]. Mathématiquement, elle mesure le cosinus de l'angle entre deux vecteurs projetés dans un espace multidimensionnel

4.6 Évaluation des approches proposées

Afin d'évaluer nos différents systèmes, les métrique suivantes ont été utilisées :
MRR@10 : le rang moyen réciproque à 10,
MAP@10 :La moyenne de la précision moyenne à 10 ,
MP@10 : Précision moyenne à 10.
MP@5 : Précision moyenne à 5.

Pour évaluer de manière adéquate nos approches, nous avons suivi le plan d'affichage suivant :

— Type de plongement
  — Indexation par document
  — Indexation par phrases

Nous évaluons dans cette section nos techniques en utilisant le base de données Arxiv+MAG.

4.6.1 Plongement avec *RoBERTa*

4.6.1.1 Indexation par document

Le tableau 4.1 montre les résultats que nous avons obtenus avec l'architecture d'indexation par document, en utilisant le plongement avec *RoBERTa*, sur les données Arxiv et MAG, et avec et sans l'expansion de la requête.

Requête			Exact MRR@10	Approximate MRR@10	Exact MAP@10	Approximate MAP@10	Exact MP@5	Approximate MP@5	Exact MP@10	Approximate MP@10
Requête initiale			0.835	0.887	0.52	0.687	0.622	0.776	0.594	0.75
Requête initiale avec définition	**Moyenne**		**0.865**	0.916	**0.547**	**0.708**	**0.654**	0.786	**0.614**	0.761
	Hybride	**alpha = 0.5 beta = 0.5**	0.839	**0.918**	0.507	0.679	0.612	0.774	0.586	0.743
		alpha = 0.7 beta = 0.3	0.863	0.904	0.532	0.704	0.64	**0.798**	0.61	**0.763**

TABLE 4.1 – Les résultats de l'évaluation de l'indexation par document avec, sans l'expansion de la requête, et avec le plongement *RoBERTa*

Selon les résultats obtenus dans le tableau 4.1, nous remarquons que les meilleurs résultats de l'évaluation "Exact" sont ceux de l'approche : expansion de la requête avec la méthode "Moyenne" avec : 0.865 pour *MRR*@10, 0.547 pour *MAP*@10, 0.654 pour *MP*@5, et 0.614 pour *MP*@10. En ce qui concerne les meilleurs résultats de l'évaluation "Approximate", ils varient entre l'expansion de la requête avec les méthodes " Moyenne " et " Hybride ". Nous constatons que pour la méthode "hybride" d'expansion de requêtes, si les paramètres alpha = 0,7 et beta = 0,3 donnent de meilleurs résultats par rapport aux paramètres alpha = 0,5 et beta = 0,5.
Nous remarquons également que les résultats les plus faibles sont ceux de l'approche sans expansion de requête(méthode de base), ce qui explique que l'expansion de requête est assez utile et augmente les performances.

4.6.1.2 Indexation par phrases

Le tableau 4.2 présente les résultats que nous avons obtenus avec l'architecture d'indexation par phrases, en utilisant le plongement avec *RoBERTa*, sur les données arXiv et MAG, et sans l'expansion de la requête.

Attribution score pour un document		Exact MRR@10	Approximate MRR@10	Exact MAP@10	Approximate MAP@10	Exact MP@5	Approximate MP@5	Exact MP@10	Approximate MP@10
Somme	**Normalisé**	**0.874**	**0.929**	0.453	0.641	0.552	0.726	0.535	0.701
	Non normalisé	0.66	0.755	0.32	0.491	0.436	0.604	0.435	0.604
Moyenne	**Normalisé**	0.868	0.916	0.453	0.64	0.554	0.728	0.536	0.701
	Non normalisé	0.842	0.904	0.469	0.654	0.582	**0.738**	0.554	0.718
Maximum	**Normalisé**	0.868	0.916	0.453	0.64	0.554	0.728	0.536	0.701
	Non normalisé	0.853	0.912	**0.471**	**0.656**	**0.586**	**0.738**	**0.555**	**0.719**

TABLE 4.2 – Les résultats de l'évaluation de l'indexation par phrases, sans l'expansion de la requête, et avec le plongement *RoBERTa*

D'après le tableau 4.2, nous observons que les meilleurs résultats, qu'ils soient "Exacts" ou "Approximatifs", proviennent de la méthode "Maximum, Non normalisé" avec : 0.471 $MAP@10_{Exact}$, 0.656 $MAP@10_{Approximate}$, 0.586 $MP@5_{Exact}$, 0.738 $MP@5_{Approximate}$, 0.555 $MP@10_{Exact}$, et 0.719 $MP@10_{Approximate}$. Par contre, la méthode "Somme, Normalisé" a obtenu les meilleurs résultats pour la mesure $MRR@10$ avec : 0.874 $MRR@10_{Exact}$, et 0.929 $MRR@10_{Approximate}$.
Nous remarquons aussi que la méthode "Somme, Non normalisé" a fourni de très mauvais résultats par rapport aux autres méthodes d'attribution du score.

4.6.1.3 Indexation par documents vs par phrases

Le tableau 4.3 présente les résultats que nous avons obtenus avec l'architecture d'indexation par phrases, en utilisant le plongement avec *RoBERTa*, sur les données Arxiv et MAG, et avec l'expansion de la requête, méthode "hybride (alpha = 0.7, beta = 0.3)".

Attribution score pour un document		Exact MRR@10	Approximate MRR@10	Exact MAP@10	Approximate MAP@10	Exact MP@5	Approximate MP@5	Exact MP@10	Approximate MP@10
Somme	**Normalisé**	0.865	**0.935**	0.513	0.708	0.634	0.792	0.59	0.76
	Non normalisé	0.68	0.793	0.365	0.546	0.48	0.648	0.486	0.659
Moyenne	**Normalisé**	0.865	**0.935**	0.511	0.708	0.636	0.794	0.588	0.76
	Non normalisé	**0.876**	0.922	**0.525**	0.716	0.644	0.798	**0.599**	0.767
Maximum	**Normalisé**	0.865	**0.935**	0.511	0.708	0.636	0.794	0.588	0.76
	Non normalisé	**0.876**	0.922	**0.525**	**0.717**	**0.648**	**0.802**	**0.599**	**0.768**

TABLE 4.3 – Les résultats de l'évaluation de l'indexation par phrases, avec l'expansion de la requête (méthode "hybride"), et avec le plongemeRoBertant

D'après les résultats présentés dans le tableau 4.3, nous réalisons que les meilleurs résultats des deux évaluations viennent de la méthode "Maximum, Non normalisé" avec : 0.876 $MRR@10_{Exact}$, 0.525 $MAP@10_{Exact}$, 0.717 $MAP@10_{Approximate}$, 0.648 $MP@5_{Exact}$, 0.802 $MP@5_{Approximate}$, 0.599 $MP@10_{Exact}$, 0.768 $MP@10_{Approximate}$.
Nous observons ici également que la méthode "Somme, Non normalisé" a donné de très faibles résultats en comparaison avec les autres méthodes d'attribution des scores.

Le tableau 4.4 montre les résultats que nous avons obtenus avec l'architecture d'indexation par phrases, en utilisant le plongement avec *RoBERTa*, sur les données Arxiv et MAG, et avec l'expansion de la requête, méthode "Moyenne".

Attribution score pour un document		Exact MRR@10	Approximate MRR@10	Exact MAP@10	Approximate MAP@10	Exact MP@5	Approximate MP@5	Exact MP@10	Approximate MP@10
Somme	**Normalisé**	0.875	0.939	0.533	0.702	**0.662**	0.81	0.602	0.75
	Non normalisé	0.831	0.913	0.523	0.693	0.632	0.784	0.605	0.757
Moyenne	**Normalisé**	0.871	0.939	0.532	0.699	0.656	0.806	0.602	0.748
	Non normalisé	0.878	0.949	0.547	0.718	**0.662**	**0.812**	0.619	**0.765**
Maximum	**Normalisé**	0.871	0.939	0.532	0.699	0.656	0.806	0.602	0.748
	Non normalisé	**0.886**	**0.957**	**0.549**	**0.719**	**0.662**	0.81	**0.62**	**0.765**

TABLE 4.4 – Les résultats de l'évaluation de l'indexation par phrases, avec l'expansion de la requête (méthode "moyenne"), et avec le plongement *RoBERTa*

D'après le tableau 4.4, nous nous réalisons que les meilleurs résultats pour les deux évaluations sont obtenus par la méthode "Maximum, Non-normalisé" avec : 0.886 $MRR@10_{Exact}$, 0.957 $MRR@10_{Approximate}$, 0.549 $MAP@10_{Exact}$, 0.719 $MAP@10_{Approximate}$, 0.662 $MP@5_{Exact}$, 0.62 $MP@10_{Exact}$, 0.765 $MP@10_{Approximate}$.
Les résultats les plus faibles proviennent de la méthode "Somme, Non normalisée".

Depuis les trois tableaux précédents (4.2, 4.3, 4.4), nous constatons que la meilleure méthode pour attribuer un score à un document est la méthode "Maximum, Non normalisé". Pour la stratégie "Somme, non normalisé", toujours donne de faibles résultats.

Le tableau 4.5 récapitule les meilleurs résultats (la méthode d'attribution du score à un document est "Maximum, Non normalisé") que nous avons obtenus avec l'architecture d'indexation par phrases utilisant le plongement *RoBERTa*, sur les données arXiv et MAG, et avec et sans expansion de requête.

Requête		Exact MRR@10	Approximate MRR@10	Exact MAP@10	Approximate MAP@10	Exact MP@5	Approximate MP@5	Exact MP@10	Approximate MP@10
Requête initiale		0.853	0.912	0.471	0.656	0.586	0.738	0.555	0.719
Requête initiale avec définition	**Hybride**	0.876	0.922	0.525	0.717	0.648	0.802	0.599	**0.768**
	Moyenne	**0.886**	**0.957**	**0.549**	**0.719**	**0.662**	**0.81**	**0.62**	0.765

TABLE 4.5 – Un tableau récapitulatif des meilleurs résultats d'évaluation de de l'indexation par phrases, et avec le plongement *RoBERTa*

Nous remarquons, suivant le tableau 4.5 que l'expansion de la requête avec la méthode "Moyenne" a conduit aux meilleurs résultats avec : 0.886 $MRR@10_{Exact}$, 0.957 $MRR@10_{Approximate}$, 0.549 $MAP@10_{Exact}$, 0.719 $MAP@10_{Approximate}$, 0.662 $MP@5_{Exact}$, 0.81 $MP@5_{Approximate}$, 0.62 $MP@10_{Exact}$.
Cela explique que l'expansion des requêtes est très utile et augmente les performances.

Le tableau 4.6 récapitule les meilleurs résultats que nous avons obtenus avec l'architecture d'indexation par documents et par phrases utilisant le plongement *RoBERTa*, sur les données arXiv et MAG, et avec et sans expansion de requête.

Type d'indexation	Requête		Exact MRR@10	Approximate MRR@10	Exact MAP@10	Approximate MAP@10	Exact MP@5	Approximate MP@5	Exact MP@10	Approximate MP@10
Indexation par document	**Requête initiale**		0.835	0.887	0.52	0.687	0.622	0.776	0.594	0.75
	Requête initiale avec définition		0.865	0.916	0.547	0.708	0.654	0.786	0.614	0.761
Indexation par phrases	**Requête initiale**		0.853	0.912	0.471	0.656	0.586	0.738	0.555	0.719
	Requête initiale avec définition	**Hybride**	0.876	0.922	0.525	0.717	0.648	0.802	0.599	**0.768**
		Moyenne	**0.886**	**0.957**	**0.549**	**0.719**	**0.662**	**0.81**	**0.62**	0.765

TABLE 4.6 – Un tableau récapitulatif des meilleurs résultats d'évaluation de toutes les approches, avec le plongement *RoBERTa*, sur les données de arXiv + MAG,

D'apprêt le tableau 4.6, nous remarquerons que :
— L'expansion de la requête donne toujours les meilleurs résultats, quelle que soit la méthode d'indexation. Ceci parce que l'utilisation de l'expansion donne plus de sens et d'explication à la requête, et cela a un impact positif parce que le plongement capture le sens et la relation entre les mots.
— La méthode d'expansion de la enquête "Moyenne" donne de plus bons résultats que la méthode "Hybride".
— La meilleure méthode d'indexation pour l'expansion de la requête est l'indexation par document. Et la meilleure dans le cas sans expansion, est l'indexation par phrases.

4.6.2 Plongement avec *SciBERT*

4.6.2.1 Indexation par document

Le tableau 4.7 montre les résultats que nous avons obtenus avec l'architecture d'indexation par phrase, en utilisant le plongement avec *SciBERT*, sur les données arXiv et MAG, et avec et sans l'expansion de la requête.

Requête			Exact MRR@10	Approximate MRR@10	Exact MAP@10	Approximate MAP@10	Exact MP@5	Approximate MP@5	Exact MP@10	Approximate MP@10
Requête initiale			0.702	**1.0**	0.227	0.984	0.352	0.99	0.308	0.989
Requête initiale avec définition	**Moyenne**		**0.856**	**1.0**	**0.477**	**0.999**	**0.62**	**1.0**	**0.555**	**0.999**
	Hybride	**alpha = 0.5 beta = 0.5**	0.789	0.995	0.286	0.991	0.424	0.998	0.373	0.993
		alpha = 0.7 beta = 0.3	0.734	0.995	0.272	0.986	0.416	0.996	0.359	0.99

TABLE 4.7 – Les résultats de l'évaluation de l'indexation par document avec et sans l'expansion de la requête avec *SciBERT*

Selon les résultats obtenus dans le tableau 4.7, nous remarquons que les meilleurs résultats de l'évaluation "Exact" et "Approximate" et sont ceux de l'approche : expansion de la requête avec la méthode "Moyenne" avec : 0.856 $MRR@10_{Exact}$, 1.0 $MRR@10_{Approximate}$, 0.477 $MAP@10_{Exact}$, 0.999 $MAP@10_{Approximate}$, 0.62 $MP@5_{Exact}$, 1.0 $MP@5_{Approximate}$, 0.555 $MP@10_{Exact}$, et 0.999 $MP@10_{Approximate}$.
Nous remarquons également que les résultats les plus faibles sont ceux de l'approche sans expansion de requête, ce qui explique que l'expansion de requête est assez utile et augmente les performances.

4.6.2.2 Indexation par phrase

Le tableau 4.8 représente les résultats que nous avons obtenus avec l'architecture d'indexation par phrase, en utilisant le plongement avec *SciBERT*, sur les données arXiv et MAG, et sans l'expansion de la requête.

Attribution score pour un document	Exact MRR@10	Approximate MRR@10	Exact MAP@10	Approximate MAP@10	Exact MP@5	Approximate MP@5	Exact MP@10	Approximate MP@10
Somme	0.544	**1.0**	0.192	0.991	0.296	0.994	0.305	0.994
Moyenne	0.801	**1.0**	**0.469**	**0.999**	**0.57**	**1.0**	**0.559**	0.999
Maximum	**0.809**	**1.0**	0.466	**0.999**	0.568	**1.0**	0.553	0.999

TABLE 4.8 – Les résultats de l'évaluation de l'indexation par phrases, avec *SciBERT*, et sans l'expansion de la requête

Selon les résultats obtenus dans le tableau 4.8, nous remarquons que les meilleurs résultats de l'évaluation "Exact" et sont ceux de l'approche "Moyenne" avec : 0.469 $MAP@10$, 0.57 $MP@5$, 0.559 $MP@10$.
En revanche, dans l'évaluation "Approximative", les résultats sont presque les mêmes, avec une variation entre 1,0 et 0,99.
Nous remarquons aussi que la méthode "Somme" a fourni de très mauvais résultats par rapport aux autres méthodes d'attribution du score.

Le tableau 4.9 récapitule les résultats (la méthode d'attribution du score à un document est "Maximum, Non normalisé") que nous avons obtenus avec l'architecture d'indexation par phrases utilisant le plongement *SciBERT*, sur les données arXiv et MAG, et avec et sans expansion de requête.

Requête		Exact MRR@10	Approximate MRR@10	Exact MAP@10	Approximate MAP@10	Exact MP@5	Approximate MP@5	Exact MP@10	Approximate MP@10
Requête initiale		0.809	**1.0**	0.466	**0.999**	0.568	1.0	0.553	**0.999**
Requête initiale avec définition	**Hybride**	0.85	**1.0**	0.502	0.998	0.618	0.998	0.584	**0.999**
	Moyenne	**0.867**	**1.0**	**0.617**	0.998	**0.72**	0.998	**0.681**	**0.999**

TABLE 4.9 – Les résultats de l'évaluation de l'indexation par phrases, avec *SciBERT*, et avec et sans l'expansion de la requête

D'après les résultats de dans le tableau 4.9, nous remarquons que les meilleurs résultats de l'évaluation "Exact" et sont ceux de l'approche : l'expansion de la requête, méthode "Moyenne" avec : 0.867 $MRR@10$, 0.617 $MAP@10$, 0.72 $MP@5$, 0.681 $MP@10$.

Par contre au l'évaluation "Approximatifs", les résultats sont presque les même, avec une variation entre 1.0 et 0.998 Nous remarquons également que la méthode sans expansion de requête a donné de moins bons résultats que les autres méthodes.

Le tableau 4.10 récapitule les meilleurs résultats que nous avons obtenus avec l'architecture d'indexation par document et par phrase utilisant le plongement *SciBERT*, sur les données arXiv et MAG, et avec et sans expansion de requête.

Type d'indexation	Requête	Exact MRR@10	Approximate MRR@10	Exact MAP@10	Approximate MAP@10	Exact MP@5	Approximate MP@5	Exact MP@10	Approximate MP@10
Indexation par document	**Requête initiale**	0.702	**1.0**	0.227	0.984	0.352	0.99	0.308	0.989
	Requête initiale avec définition	0.856	**1.0**	0.477	**0.999**	0.62	**1.0**	0.555	**0.999**
Indexation par phrases	**Requête initiale**	0.809	**1.0**	0.466	**0.999**	0.568	**1.0**	0.553	**0.999**
	Requête initiale avec définition	**0.867**	**1.0**	**0.617**	0.998	**0.72**	0.998	**0.681**	**0.999**

TABLE 4.10 – Un tableau récapitulatif des meilleurs résultats d'évaluation de toutes les approches avec *SciBERT*

D'après le tableau 4.10, nous pouvons déduire que :
— Avec l'expansion de la requête, les résultats sont toujours meilleurs. Cela est dû au fait que l'utilisation de l'expansion donne plus de sens et d'explication à la requête.
— La méthode d'expansion de requête "Moyenne" donne de meilleurs résultats que la méthode "Hybride".
— L'indexation par phrases est la meilleure.

4.7 Création d'un nouveau corpus d'évaluation

4.7.1 Présentation de la base de données de l'ACM

Notre travail étant basé sur la recherche d'experts, nous avons voulu avoir la meilleure représentation de l'expertise de chaque auteur dans notre base. Lors de notre collecte de données, nous avons constaté, que ACM Digital Library, présentait des données de hautes qualités, c'est à dire que les articles étaient *peer-reviwed*, et les domaines des auteurs étaient très réalistes. De plus, notre sélection des auteurs était aléatoire, ce qui permet d'éviter de tomber dans des biais tels que travailler seulement avec des auteurs très prolifiques ou au contraire n'ayant pas assez de documents. En d'autres termes nous avons essayé d'avoir un échantillon assez large qui reste représentatif d'un cas d'utilisation réel ou les auteurs sont très, moyennement et moins prolifiques.
Pour chaque auteur nous avons extrait en utilisant le *web scraping* directement du site d'ACM, tous les résumés (abstracts) de ses articles(291 811 articles), le nombre de citations, les mots clés de ses articles, ses sujets de recherches, ainsi que d'autres données. Le résumé sera utilisé pour l'indexation des documents, l'information sur les domaines de recherches sera utilisée pour l'évaluation des résultats.

	ACM	arXiv + MAG
Median	14	2
Moyenne	27.7835	5.6407
Q1	5	1
Q3	35	5
Maximum	1	1
Maximum	586	279

TABLE 4.11 – Des statistiques sur le nombre d'articles pour chaque auteur, des bases de données ACM et Arxiv+MAG

Tableau 4.11, un tableau de statistiques, a le but de comparer la base de données ACM collectée, et la base de données Arxiv+MAG.

4.7.2 Collecte et nettoyage des données

4.7.2.1 La collecte des données

Afin de collecter la nouvelle base de données, nous avons utilisé une technique appelée "*Web Scraping*".

Le **web scraping**, également connu sous le nom d'extraction ou de récolte du Web, est une technique permettant d'extraire des données du *World Wide Web* (WWW) et de les sauvegarder dans un système de fichiers ou une base de données pour les récupérer ou les analyser ultérieurement. En général, les données du Web sont extraites à l'aide du protocole de transfert hypertexte (HTTP) ou d'un navigateur Web. Cette opération est réalisée soit manuellement par un utilisateur ou automatiquement par un robot ou un *crawler web*[120].

4.7.2.2 Le nettoyage des données

Le nettoyage des données consiste à corriger des problèmes ou des erreurs systématiques dans des données désordonnées. Le nettoyage de données le plus utile implique une expertise approfondie du domaine et peut consister à identifier et à traiter des observations spécifiques qui peuvent être incorrectes. Il existe de nombreuses raisons pour lesquelles les données peuvent avoir des valeurs incorrectes, comme les erreurs de frappe, la corruption, les doublons, etc.

Il existe des opérations générales de nettoyage des données qui peuvent être effectuées, telles que :

— L'utilisation de statistiques pour définir les données normales et identifier les valeurs aberrantes.
— Identifier les colonnes qui ont la même valeur ou aucune variance et les supprimer.
— Identifier les lignes de données en double et les supprimer.
— Marquer les valeurs vides comme manquantes.
— Imputation des valeurs manquantes à l'aide de statistiques ou d'un modèle appris[121].

Nous avons effectué le nettoyage des données sur notre base de données de la manière suivante :

Tout d'abord, nous avons supprimé tous les doublons des articles et des auteurs. Ensuite, nous avons fusionné les données des auteurs qui sont la même personne mais qui ont une forme de nom différente, exemple : "**Jason Brownlee**" et "**J. Brownlee**". Puis, nous avons supprimé tous les auteurs qui n'ont pas de tags, et les articles qui ne possèdent pas de résumés. Et enfin, nous avons nettoyé le titre et le résumé des articles, en supprimant les ponctuations, les espaces supplémentaires, les adresses email et les URLs, en transformant les nombres en forme alphabétique (en utilisant la bibliothèque num2words de Python), et en divisant le résumé en liste de phrases (la division était au niveau des points ".").

Problème rencontré :

Après l'acquisition des données en utilisant le web scraping, nous avons rencontré un sérieux problème qui pouvait nuire à la qualité de notre travail. Plusieurs mots qui figuraient dans le résumé de la plupart des articles étaient collés. exemple : "*imagesegmentationis*", ce qui devrait être "*image segmentation is*".

Nous avons résolu totalement ce problème en utilisant l'API *Wiktionary* , avec la loi de *Zipf*[5].

La loi de *Zipf* utilise un dictionnaire sur les statistiques des mots pour séparer un mot collé. Le principe c'est que si ce mot figure déjà dans ce dictionnaire il ne sera pas modifié, sinon nous cherchons à trouver les deux ou plusieurs mots qui le forme. De ce fait, si un mot est correct et ne figure pas dans ce dictionnaire, il sera séparé par erreur. Cette loi utilisée seule a pu obtenir une précision acceptable. D'après nos observations, près de 70% des mots corrigés étaient effectivement faux. Pour une meilleure précision, nous avons décidé d'utiliser un dictionnaire plus large, afin de vérifier l'existence du mot avant d'entreprendre à le corriger. Après avoir effectué plusieurs recherches, nous avons conclu qu'utiliser *wiktionary* était notre meilleure option.

wiktionary est un projet de la *Wikimedia Foundation* dont le but est de définir tous les mots de toutes les langues. Une API est disponible dans Python[123]. L'API envoie une requête HTTP avec un mot en entrée, et renvoie un fichier JSON comme réponse. Si ce mot recherché figure dans la base de données de *wiktionary*, le résultat sera la définition du mot, sinon ça sera une liste vide. De cette manière, nous avons un excellent indicatif sur l'existence d'un mot donné, et ainsi éviter d'utiliser la loi de *Zipf*, avec des mots corrects.
Nous avons de cette manière atteint une précision de 100% vis-à-vis de la validité des mots corrigés. Néanmoins, l'API envoyait des requêtes pour chaque mot de notre corpus, ce qui prenait beaucoup de temps. Pour remédier à ceci nous vérifions d'abord l'existence du mot dans un dictionnaire contenant 400,000 mots courant de la langue Anglaise, la recherche en local prend beaucoup moins de temps qu'une requête web, pour encore plus d'optimisation, nous avons construit un arbre binaire de recherche qui contient les 400,000 mots du dictionnaires.

4.7.3 Échantillonnage pondéré pour la sélection des requêtes

Dans le but de sélectionner l'ensemble des requêtes utilisées dans l'évaluation de la base de données d'ACM, et pour s'assurer que cet ensemble est assez représentatif de la population, nous avons appliqué la méthode de l'échantillonnage pondéré. Cette méthode est un échantillon dans lequel chaque unité d'échantillonnage s'est vue attribuer un poids à utiliser dans une ana-

5. La loi de Zipf établit une relation entre le rang des mots d'un texte ordonnées par ordre décroissant de fréquences d'apparition et cette fréquence. Elle a pris le nom de son auteur, *George Kingsley Zipf* [122]

lyse ultérieure[124].

Nous avons adapté le principe de la méthode d'échantillonnage pondéré comme suit : nous avons récupéré tous les tags de tous les auteurs, puis nous avons assigné une probabilité à chaque tag (probabilité d'un tag = fréquence d'un tag / le nombre de tous les tags), ensuite nous avons employé la fonction "random.choices" de la bibliothèque "random" de python[125] pour sélectionner l'ensemble des requêtes. Cet ensemble est disponible dans le tableau en annexe.

4.7.4 Évaluation des approches proposées avec le nouveau corpus

4.7.4.1 Sans le domaine de l'auteur pour la distribution du score

Le tableau 4.12 résume nos résultats relatifs à la base de données ACM, avec indexation par document et par phrase en utilisant le plongement *SciBERT*, et avec et sans expansion de requête. Nous avons utilisé les meilleurs paramètres obtenues avec la base de données Arxive-MAG ("Maximum, non normalisée" pour l'attribution score pour un document, la methode "moyenne" pour l'expansion de la requête).

Type d'indexation	Requête	Exact MRR@10	Approximate MRR@10	Exact MAP@10	Approximate MAP@10	Exact MP@5	Approximate MP@5	Exact MP@10	Approximate MP@10
Indexation par document	**Requête initiale**	0.713	1.0	0.346	**1.0**	0.496	**1.0**	0.443	**1.0**
	Requête initiale avec définition	**0.757**	1.0	**0.428**	0.998	**0.578**	0.998	**0.533**	0.999
Indexation par phrases	**Requête initiale**	0.671	**1.0**	0.272	0.997	0.404	0.998	0.368	0.998
	Requête initiale avec définition	0.726	**1.0**	0.35	**1.0**	0.492	**1.0**	0.457	**1.0**

TABLE 4.12 – Les résultats d'évaluation de toutes les approches avec la base de données ACM, sans le domaine de l'auteur pour la distribution du score

D'après les résultats de dans le tableau 4.12, nous remarquons que les meilleurs résultats de l'évaluation "Exact" sont ceux de l'approche : indexation par document avec l'expansion de la requête, méthode "Moyenne" avec : 0.757 $MRR@10$, 0.428 $MAP@10$, 0.578 $MP@5$, 0.533 $MP@10$.
Par contre au l'évaluation "Approximatifs", les résultats varient entre 0.997 et 1.0, pour toutes les métriques, et quel que soit la configuration. On remarque également, que l'expansion de la requête améliore la qualité des résultats pour chaque type d'indexations.

4.7.4.2 Avec le domaine de l'auteur pour la distribution du score

Le tableau 4.13 récapitule les résultats obtenus en utilisant la base de données ACM, avec l'indexation par documents et par phrases en utilisant le plongement *SciBERT*, avec et sans expansion de requête, et l'utilisation de domaine de l'auteur pour la distribution du score. Nous avons utilisé les meilleurs paramètres obtenues avec la base de données Arxive-MAG ("Maximum, non normalisée" pour l'attribution score pour un document, la methode "moyenne" pour l'expansion de la requête).

Type d'indexation	Requête	Exact MRR@10	Approximate MRR@10	Exact MAP@10	Approximate MAP@10	Exact MP@5	Approximate MP@5	Exact MP@10	Approximate MP@10
Indexation par document	**Requête initiale**	0.71	**1.0**	0.314	0.995	0.446	0.996	0.424	0.997
	Requête initiale avec définition	**0.803**	**1.0**	**0.422**	**1.0**	**0.566**	**1.0**	**0.526**	**1.0**
Indexation par phrases	**Requête initiale**	0.686	**1.0**	0.332	0.999	0.476	**1.0**	0.433	0.999
	Requête initiale avec définition	0.689	**1.0**	0.237	0.988	0.362	0.994	0.341	0.992

TABLE 4.13 – Les résultats d'évaluation de toutes les approches avec la base de données ACM, avec le domaine de l'auteur pour la distribution du score

De tableau 4.13, nous remarquons que les meilleurs résultats de l'évaluation sont ceux de l'approche : indexation par document avec l'expansion de la requête, méthode "Moyenne" avec : 0.803 $MRR@10_{Exact}$, 0.803 $MRR@10_{Approximate}$, 0.422 $MAP@10_{Exact}$), 1.0 $MAP_{Approximate}@10$, 0.566 $MP@5_{Exact}$, 1.0 $MP@5_{Approximate}$, 0.526 $MP@10_{Exact}$, 1.0 $MP@10_{Approximate}$.Les resultats de l'évaluation "Approximate", varient entre 0.988 et 1.0, quel que soit la configuration.

Nous remarquons que l'expansion de requête améliore la qualité des résultats pour chaque type d'indexation.

Pour les types d'indexation, nous pouvons voir que sans expansion de requête, l'indexation par phrase est la meilleure, mais avec l'expansion de requête, l'indexation par document la surpasse.

4.8 Discussion générale

Dataset	Type de plongement	Type d'indexation	Requête		Exact MRR@10	Approximate MRR@10	Exact MAP@10	Approximate MAP@10	Exact MP@5	Approximate MP@5	Exact MP@10	Approximate MP@10
arXiv + MAG	RoBerta	Indexation par document	Requête initiale		0.835	0.887	0.52	0.687	0.622	0.776	0.594	0.75
			Requête initiale avec définition		0.865	0.916	0.547	0.708	0.654	0.786	0.614	0.761
		Indexation par phrases	Requête initiale		0.853	0.912	0.471	0.656	0.586	0.738	0.555	0.719
			Requête initiale avec définition		**0.886**	0.957	0.549	0.719	0.662	0.81	0.62	0.765
	SciBert	Indexation par document	Requête initiale		0.702	**1.0**	0.227	0.984	0.352	0.99	0.308	0.989
			Requête initiale avec définition		0.856	**1.0**	0.477	**0.999**	0.62	**1.0**	0.555	**0.999**
		Indexation par phrases	Requête initiale		0.809	**1.0**	0.466	**0.999**	0.568	**1.0**	0.553	**0.999**
			Requête initiale avec définition		0.867	**1.0**	**0.617**	0.998	**0.72**	0.998	**0.681**	**0.999**
ACM	SciBert	Indexation par document	Requête initiale	Sans le domaine de l'auteur pour la distribution du score	0.713	**1.0**	0.346	**1.0**	0.496	**1.0**	0.443	**1.0**
				Avec le domaine de l'auteur pour la distribution du score	0.71	**1.0**	0.314	0.995	0.446	0.996	0.424	0.997
			Requête initiale avec définition	Sans le domaine de l'auteur pour la distribution du score	0.757	**1.0**	**0.428**	0.998	**0.578**	0.998	**0.533**	0.999
				Avec le domaine de l'auteur pour la distribution du score	**0.803**	**1.0**	0.422	**1.0**	0.566	**1.0**	0.526	**1.0**
		Indexation par phrases	Requête initiale	Sans le domaine de l'auteur pour la distribution du score	0.699	**1.0**	0.328	0.999	0.472	**1.0**	0.426	0.999
				Avec le domaine de l'auteur pour la distribution du score	0.686	**1.0**	0.332	0.999	0.476	**1.0**	0.433	0.999
			Requête initiale avec définition	Sans le domaine de l'auteur pour la distribution du score	0.69	**1.0**	0.229	0.992	0.362	0.996	0.334	0.995
				Avec le domaine de l'auteur pour la distribution du score	0.689	**1.0**	0.237	0.988	0.362	0.994	0.341	0.992

TABLE 4.14 – Un tableau récapitulatif de tous les résultats des approches proposées avec les deux bases de données, Arxiv+MAG et ACM

Dans notre travail nous avons appliqué deux différentes variante de modèle *BERT*, a savoir *RoBERTa* et *SciBERT*. L'objective était de voir si *SciBERT* qui est entraîné sur des données scientifique est plus performent que *RoBERTa* apparu avant lui. Nous avons proposer d'indexer un document par ces phrases, et comparer cette approche avec l'indexation d'un document en entier, la motivation était de voir si travailler directement avec les phrases améliore les résultats. Nous avons aussi évalué deux approches pour l'expansion de la requête, a savoir, l'approche "Moyenne", et l'approche "hybride".

Lors de cette partie nous allons d'abord analyser les résultats obtenus en évaluant nos différentes configurations sur la base de données Arxiv+MAG, puis discuter ceux obtenus avec

ACM (Le tableau 4.14 présente toutes ces évaluations). Concernant l'indexation par phrases avec Arxiv+MAG, c'est la stratégie maximum d'attribution de score pour un document, sans normalisation, qui fut la meilleure, ceci veut dire que travailler avec les phrases les plus significatives d'un document a été selon nos évaluations la meilleure manière. L'expansion de la requête la plus performante fut l'expansion moyenne. Cette méthode donne presque à chaque fois, une meilleure performance que la méthode "hybride", et cela peut être dû au fait que dans la méthode "hybride", les scores des articles pertinents de la requête initiale et de la définition de la requête, doivent être normalisés. Car il se peut qu'un article soit pertinent pour la requête initiale, mais pas pour la définition de la requête. Et cela pourrait diminuer les résultats des évaluations.

Les testes réalisés sur la base de données d'Arxiv+MAG ont montré que les meilleurs résultats sont obtenus avec la configuration utilisant l'indexation par phrases avec *SciBERT*, et avec expansion "moyenne" de la requête, dépassant ainsi l'état de l'art sur cette base de données.

Toutes les configurations utilisées avec Arxiv+MAG ont obtenu un $MRR@10_{approximate,uniform}$ d'au moins 0.809, sauf le modèle de base avec *SciBERT*, qui a étrangement trouvé des difficultés avec cette métrique (0.702). Néanmoins, Nous pouvons remarquer que le modèle de base a obtenu 0.835 avec $MRR@10_{approximate,uniform}$, donc la même métrique, en utilisant le plongement par *RoBERTa*, ce qui signifie que le changement de la méthode de plongement a eu un impact négatif. Le faible score $MRR@10_{approximate,uniform}$ mentionné précédemment fut augmenté jusqu'à 0.856, en utilisant l'expansion "moyenne" de la requête. L'un de nos constat majeur est que l'expansion de la requête améliore la qualité de tous les résultats et est bénéfique pour tous nos systèmes utilisés. Une autre constatation importante concerne l'indexation par phrase avec *RoBERTa*, ou nous avons obtenu de façon générale de meilleures scores de *MRR* mais de moins bons scores en précisions, par rapport à l'indexation par document. Nous pouvons tenter d'expliquer ceci par le fait que le *MRR* détermine si le premier expert est pertinent ou non, et il est possible que le système trouve le premier auteur pertinent mais que les autres, ou certains d'entre eux, ne le soient pas. Cela pourrait expliquer pourquoi dans certains cas, les résultats du *MRR* sont élevés, mais les métriques de précision sont faibles. L'utilisation de l'expansion de la requête a comblé ce gap et a permis d'améliorer la précision de l'indexation par phrase. Une explication plausible est que l'utilisation de l'expansion de la requête permet de trouver plus de documents pertinents lors de la phase de recherche, ce qui permettrait de sélectionner plus d'auteurs pertinents dans la liste des résultats finaux. Toujours avec la base de données Arxiv+MAG, nous remarquons qu'avec *SciBERT*, l'indexation par phrases a été en général meilleure que l'indexation par documents. Ceci est peut être dû à la nature de *SciBERT* qui, étant entraîné sur des données scientifiques est donc plus proche des documents que nous avons utilisés, et de ce fait modéliserait mieux que *RoBERTa* la sémantique d'une phrase. Ceci pourrait avoir eu pour effet, non seulement de sélectionner de meilleures phrases pour attribuer un score à l'auteur (améliorer la précision de l'indexation par phrases) mais un peu l'effet inverse dans l'indexation par documents, car le fait de bien capturer la sémantique des différentes phrases d'un document, les phrases les moins significatives dans ce document auront plus d'impact dans la représentation finale (la moyenne des phrases). Nous pouvons également observer le même phénomène avec *RoBERTa* dont l'amélioration des résultats par l'expansion de la requête fut aussi observée avec *SciBERT*

Au final, pour comparer entre *SciBERT* et *RoBERTa*, Nous voulons d'abord souligner que durant toutes les évaluations effectuée *approximate* a donné toujours de meilleurs résultats que exact, cela est due au fait que *approximate* utilise un seuil (0.7 durant les évaluations) afin de vérifier si la requête appartient au tags de l'expert pour lui considérer comme pertinent, si ce seuil est égale a 1, alors cela revient à évaluer avec la méthode *exact*, qui est donc de trouver l'exacte ressemblance entre la requête et un des tag de l'auteur pour le considérer comme

pertinent, ce qui est plus difficile. Pour revenir a la comparaison entre *SciBERT* et *RoBERTa*, nous notons que ce dernier, a eut de meilleurs résultats dans les évaluations *Exact*, et *SciBERT* fut meilleur avec *approximate*. Cela veut dire que *RoBERTa* a été plus performant à trouver les auteurs dont les tags correspondent exactement a la requête. Les auteurs ayant des tags correspondants exactement a la requête sont considérés comme les plus pertinents. Donc, *RoBERTa* a été plus performant que *SciBERT* avec la base de données Arxiv-MAG. En revanche un meilleur score *approximate* pour *SciBERT* peut s'expliquer par le fait qu'il calcul des plongements pour plus proches à la requête, ceci est possible car il est plus familier avec les termes scientifiques.

Dans la deuxième phase des évaluations c'est la base de données ACM qui a été utilisée, cette base nous a permis d'introduire le domaine de l'auteur pour calculer le score. Nous avons constaté que cela a apporté quelque améliorations de la précision dans certain cas. Nous avons en outre obtenu le meilleur score *MRR@10* en l'utilisant avec l'indexation par documents et l'expansion de la requête. Néanmoins dans la majorité des cas nous n'avons pas eu d'amélioration remarquable, peut être parce que la modification apportée a la formule de vote, ne change beaucoup le score final, qui sera proche du cas sans utilisation du domaine de l'auteur. On peut remarquer également que de façon générale l'indexation par documents était meilleure que l'indexation par phrases sur la base de données ACM. Les améliorations apportées par l'expansion de la requête sont confirmées dans le cas de l'indexation par documents. Néanmoins une régression dans la qualité des résultats en utilisant l'indexation par phrases est remarquées, ainsi qu'une légère dégradation causée par l'expansion de la requête, cela est peut être du au paramètre empirique k qui détermine le nombre de phrases a extraire. Peut être en augmentant le nombre de phrases sur lesquelles nous nous basons pour calculer le score améliorera les résultats, ou alors que l'indexation par phrases n'est pas efficace avec des bases de données volumineuses.

4.9 Conclusion

Ce chapitre presentait les informations sur les environnements matériels et logiciels utilisés pour réaliser nos approches. Il presentait aussi les bases de données utilisées pour évaluer notre stratégie. Enfin, nous avons présenté les méthodes utilisées pour l'évaluation, qui sont "Exact" et "Approximate". Nous avons ensuite commencé à présenter les évaluations de nos approches, d'abord avec le dataset Arxiv+MAG, puis avec le dataset ACM. Enfin, nous avons présenté une section dans laquelle nous avons discuté les résultats de toutes les évaluations, en définissant les meilleures et les pires approches.

Dans le prochain chapitre, nous présenterons notre site web pour la recherche d'experts, avec des chercheurs algériens.

Chapitre 5

Un portail pour la recherche d'experts académiques algériens

5.1 Introduction

Suite à la réalisation et à l'évaluation des approches proposées conçues dans le chapitre précédent, nous arrivons dans ce chapitre à la mise en œuvre de notre portail de la recherche d'experts académiques algériens. Ce portail nous permettra d'utiliser les approches proposées avec des spécialistes et experts algériens.
Dans la première partie de ce chapitre, nous présentons les outils et le langage de développement employés pour la réalisation du site web. Nous présentons par la suite les différentes pages web de notre site. À la fin, nous démontrons toutes les fonctionnalités de notre site web, en l'illustrant par des captures d'écran et des explications d'un exemple. Le code source est disponible sur : `https://github.com/Serine07/PFE-Web-Site`.

5.2 Outils et langage de développement

Nous avons développé le site web sur Spyder en utilisant Python et de nombreuses bibliothèques comme mentionné ci-dessus dans 4.2.2. En plus de cela, de plusieurs autres langages et outils sont utilisés pour le réaliser, qui sont :

HTML (*Hyper Text Markup Language*) est le langage de balisage standard pour créer et décrire la structure des pages Web. Il est composé d'une série d'éléments, qui indiquent au navigateur comment afficher le contenu. Il est supporté et développé par W3C[126]. HTML5 est la dernière révision majeure de HTML, cette version a été finalisée en 2014.

CSS (*Cascading Style Sheets*) est le langage à utiliser pour styliser une page Web. Il décrit comment les éléments HTML doivent être affichés à l'écran, sur papier ou sur d'autres supports. Les feuilles de style CSS permettent d'économiser beaucoup de travail, car elles peuvent contrôler la mise en page de plusieurs pages Web en une seule fois. Les feuilles de style externes sont stockées dans des fichiers CSS[127].

JavaScript est un langage de programmation orienté prototype, créé en 1995 par la *Netscape Communication Corporation*. Il est principalement utilisé sur Internet, à côté du HTML et du CSS. Il utilise des scripts pour créer du contenu dynamique, il peut stocker des valeurs, effectuer des opérations ou exécuter du code en fonction de certains événements. JavaScript peut être utilisé soit du côté du serveur, soit du côté du client. Il est apprécié pour sa simplicité, sa flexibilité et sa puissance[128].

Flask est un framework d'application web WSGI [1] léger. Il est conçu pour permettre une prise en main rapide et facile, tout en offrant la possibilité d'évoluer vers des applications complexes. Il a commencé comme une simple enveloppe autour de Werkzeug [2] et Jinja et est devenu l'un des frameworks d'applications web Python les plus populaires[129].

Jinja est un moteur de création de modèles rapide, expressif et extensible. Des espaces réservés dans le modèle permettent d'écrire du code similaire à la syntaxe Python. Ensuite, le modèle reçoit des données pour rendre le document final[130].

.

5.3 Présentation du site web

Afin de bien illustrer notre travail, nous avons mis en œuvre un site web destiné aux experts algériens de différentes universités du pays.
La base de données utilisée provient de deux sources. La première est la bibliothèque numérique de l'ACM, et la seconde est OpenAlex [3]. Nous avons fusionné les deux afin d'avoir une base de données assez complète. Le résultat de cette fusion est un total de 8 748 experts, et 14 780 articles.
Nous avons décidé de choisir la langue anglaise pour notre site web car, la totalité des données sont en anglais (articles, affiliation des auteurs, etc.).
Ce site web est composé de plusieurs pages, dont les suivantes :

5.3.1 Page d'accueil

La page d'accueil contient principalement une barre de recherche, pour une recherche directe, l'utilisateur n'a qu'à entrer sa requête.Pour une recherche avancée, l'utilisateur doit appuyer sur le bouton "Advaced search" afin d'être redirigé vers la page web de la recherche avancée.
En bas de la page, trois experts suggérés sont listés, et pour obtenir plus d'informations sur eux, l'utilisateur doit appuyer sur le bouton "More details" (voir figure 5.1).

1. Web Server Gateway Interface
2. Werkzeug est une bibliothèque complète d'applications web WSGI
3. OpenAlex est un catalogue ouvert et complet d'articles savants, d'auteurs et d'institutions. Il s'agit d'un index de centaines de millions d'entités interconnectées dans le système de recherche mondial. OpenAlex est un ensemble de données à source ouverte.

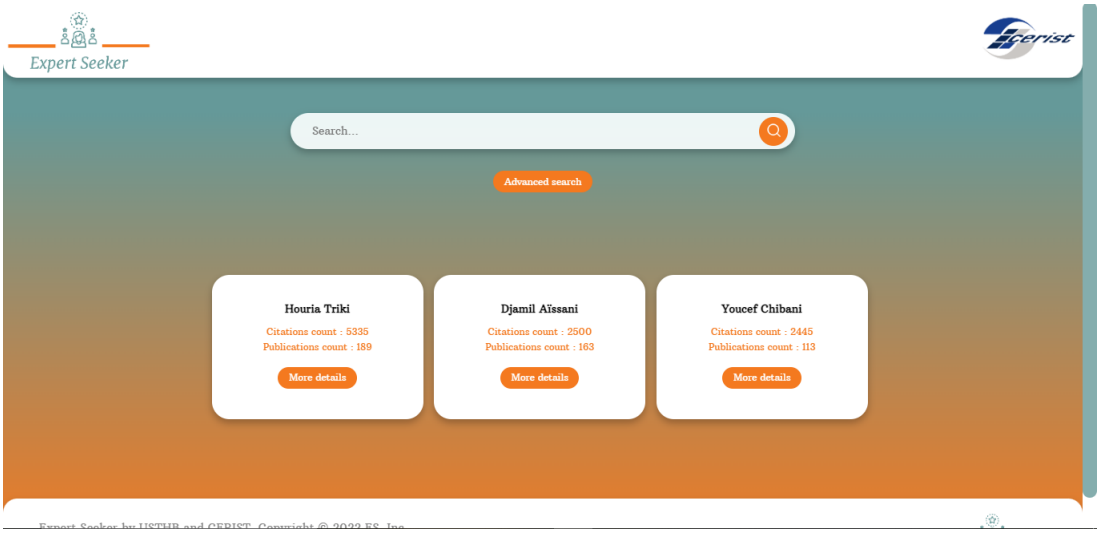


FIGURE 5.1 – Page d'accueil

5.3.2 Page de recherche avancée

La page de recherche avancée contient une barre de recherche pour saisir la requête (une liste de suggestions apparaîtra dès que l'utilisateur commencera à écrire sa requête. Cette liste est classé par ordre alphabétique.), ainsi qu'un ensemble de listes déroulantes et une roulette pour sélectionner les différents paramètres de la recherche(voir figure 5.2). Ces paramètres sont :

- **Indexation type** : ceci permet de sélectionner le type d'indexation (par phrases ou par document).
- **With or without definition** : permet de définir si l'expansion de la requête est utilisée ou non.
- **Embedder type** : permet de sélectionner le type de plongement (RoBERTa ou Sci-BERT).
- **University** : permet de sélectionner l'université dans laquelle les experts des résultats doivent travailler.
- **Score distribution** : permet de choisir si nous attribuons le score complet a tous les auteurs, ou nous utilisons le domaine de l'auteur pour le faire.
- **Results number** : et enfin, une roulette pour choisir le nombre de résultats.

Et en bas de la page, se trouvent deux boutons, l'un pour réinitialiser les paramètres de recherche avancée "Reset", et l'autre pour lancer la recherche "Search".

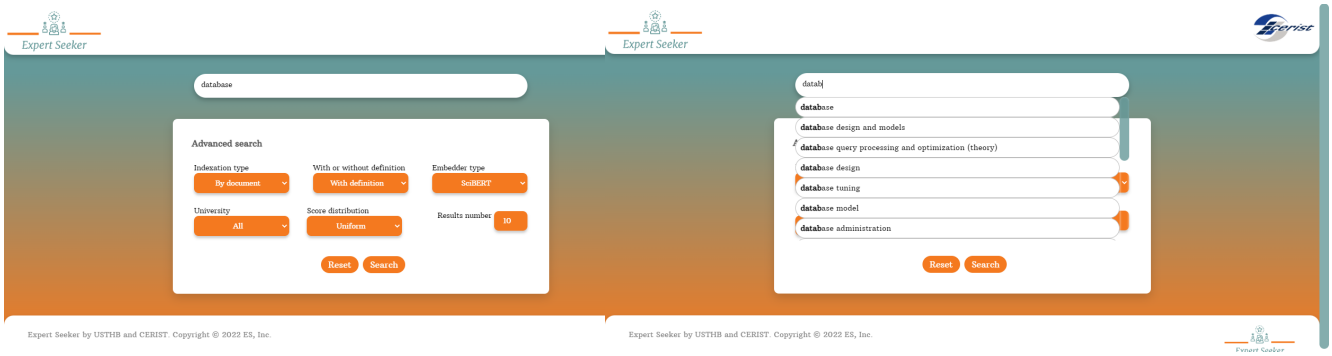


FIGURE 5.2 – Page de la recherche avancée

5.3.3 Page des résultats de la recherche

La page de résultats de recherche affiche les N premiers experts sous la forme d'un ensemble de fiches, qui contiennent le nom de l'expert, son score, son affiliation, le nombre de ses publications et ses tags (voir Figure 5.3). En cliquant sur le bouton "more details", une page présentant toutes les informations relatives à l'expert apparaît.

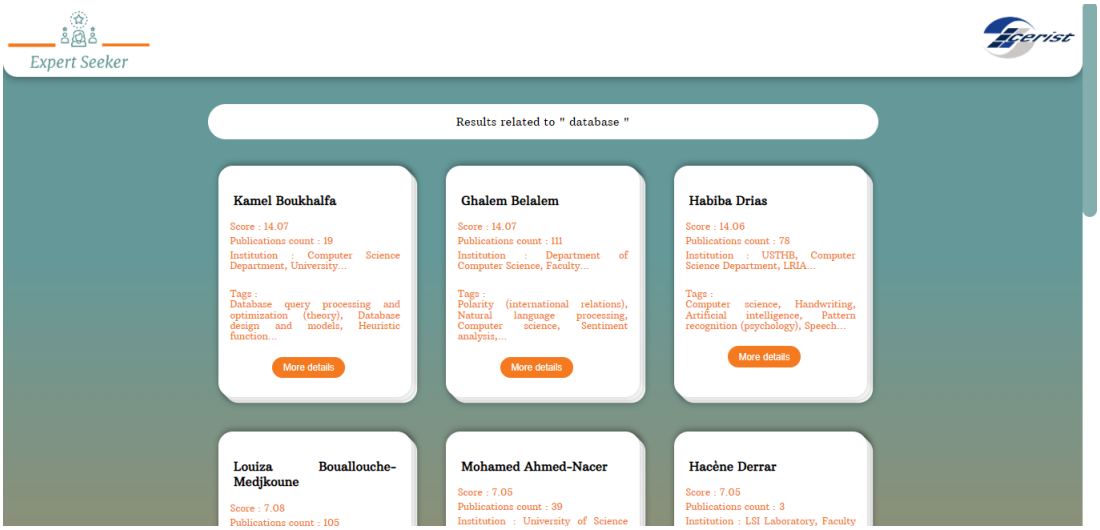


FIGURE 5.3 – Page des résultats de la recherche

5.3.4 Page des informations de l'expert

La page d'information de l'expert contient toutes les informations relatives à l'expert(voir figure 5.4).
En haut de la page, nous trouvons le nom de l'expert, son nombre de publication, son affiliation, et ses tags. En dessous, une liste de fiches, dont chaque fiche présente des informations sur l'un de ces articles (titre, noms des auteurs, et le résumé). En haut de cette liste, apparaissent les articles ayant une relation avec la requête, et en bas, les articles restants de l'auteur.

FIGURE 5.4 – Page des informations de l'expert

Chaque fiche du résumé d'un article contient les informations suivantes : le titre, le nom des auteurs, et le résumé(voir figure 5.5).

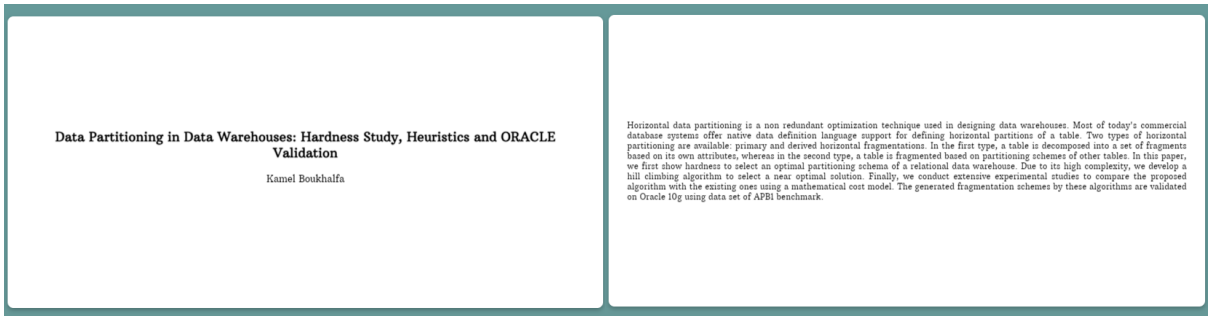


FIGURE 5.5 – Fiche d'un résumé d'un article

5.4 Exemple de recherche dans notre site web

Pour cet exemple de recherche avancée, nous commençons par saisir la requête, qui est "query expansion", puis nous choisissons les paramètres de la recherche comme suit : indexation par document, avec définition, plongement avec SciBERT, toutes les universités, distribution de score est uniforme, et le nombre de résultats est 10. Ensuite, nous appuyons sur le bouton "Search" pour lancer la recherche. La figure 5.6 illustre cette étape.

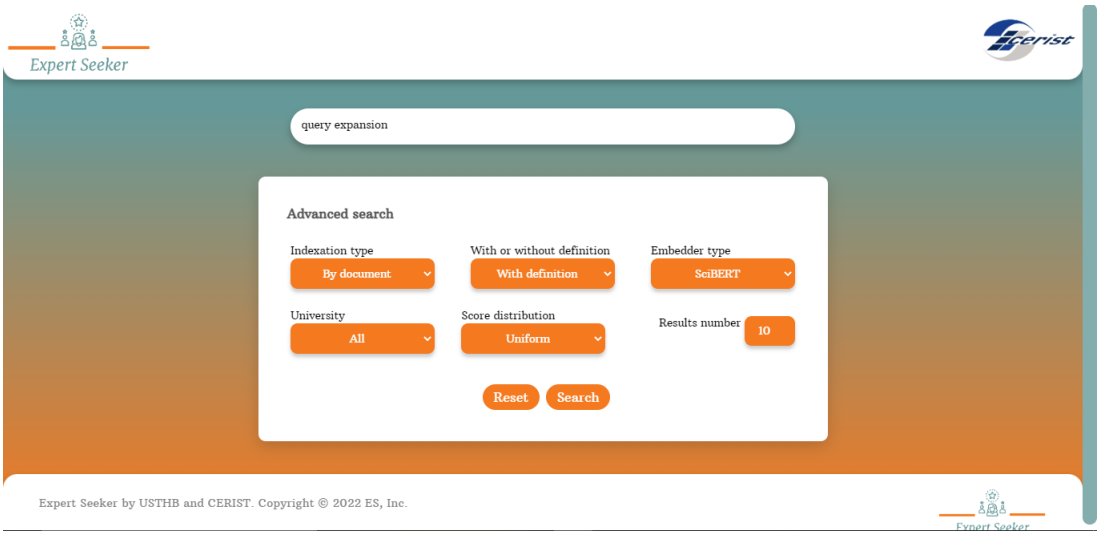


FIGURE 5.6 – Page des résultats de la recherche

Après avoir lancé la recherche, un ensemble de 10 meilleurs experts apparaît sur l'interface(voir figure 5.7), classés en fonction du score de chaque expert par rapport à la requête.Comme nous pouvons le constater, les tags des meilleurs experts sont en rapport avec la requête, qui est " query expansion".

Les trois meilleurs experts sont : Habiba Drias de l'USTHB, Ghalem Belalem de l'université d'Oran 1 Ahmed Ben Bella , et Ilyes Khennak de l'USTHB.

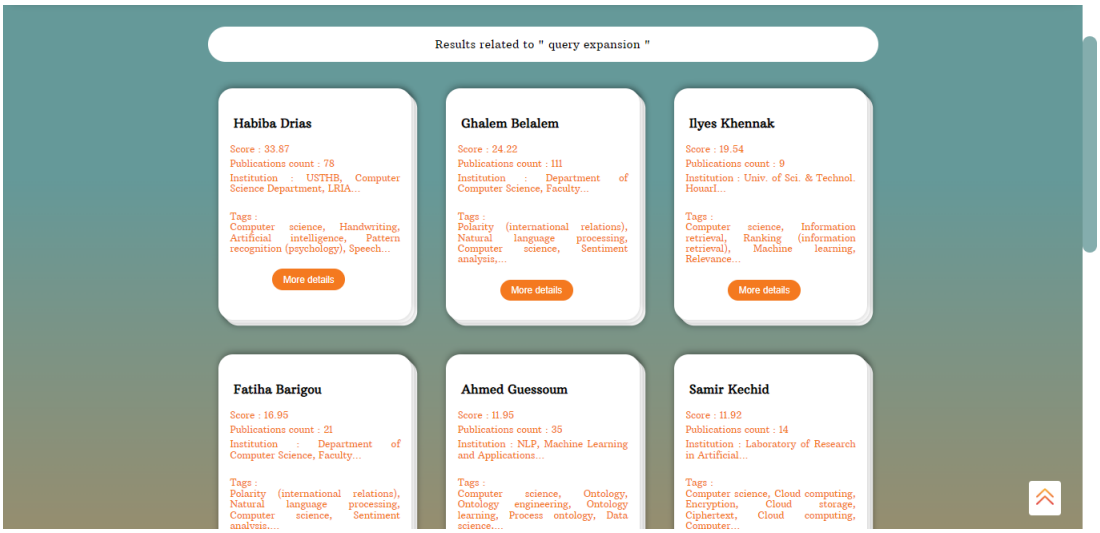


FIGURE 5.7 – Page de la recherche avancée

Maintenant nous consultons le profil de l'expert le avec le plus haut score (33.87) dans la liste des résultats, qui est "Habiba Drias"(voir figure 5.8).

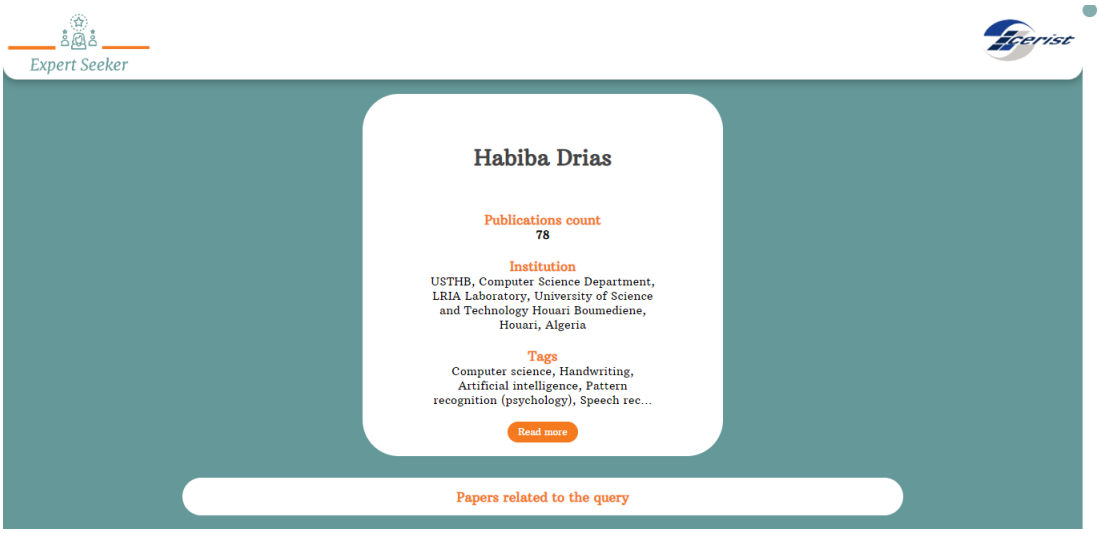


FIGURE 5.8 – Page de profil de l'expert "Habiba Drias"

Treize articles de "Habiba Drias" qui sont liés à la requête "query expansion" ont été affichés en haut de la page. Les trois premiers sont : "Data mining techniques and nature-inspired algorithms for query expansion", "Bat-inspired algorithm based query expansion for medical web information retrieval", "Clustering algorithms for query expansion based information retrieval"(voir figure 5.9). Il est clair que les trois résultats ont une relation directe avec la requête "query expansion".

FIGURE 5.9 – Page de profil de l'expert "Habiba Drias" (les articles ayant une relation avec la requête)

En bas de cette page, une liste des documents qui ont été écrits par "Habiba Drias", mais n'ayant aucun rapport avec la requête(voir figure 5.10).

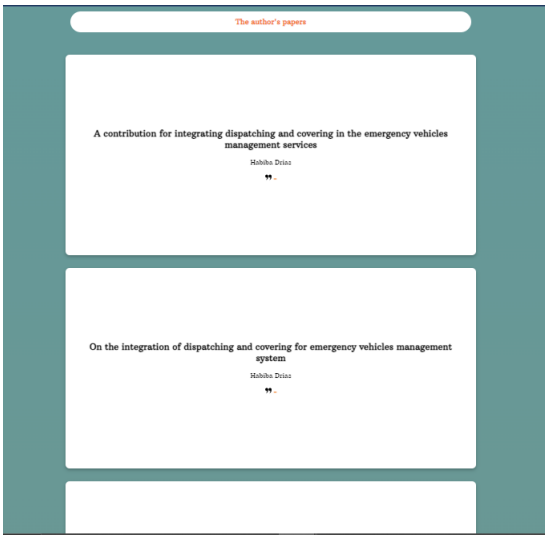


FIGURE 5.10 – Page de profil de l'expert "Habiba Drias" (les articles non liés à la requête)

5.5 Conclusion

Dans ce dernier chapitre, nous avons présenté les outils ayant servi à l'élaboration les pages web de notre site sur la recherche d'experts académiques algériens. Par la suite, nous avons présenté à l'aide de captures d'écran, une vue générale de notre site avec ces différentes fonctionnalités. Et à la fin, nous avons ajouté un exemple pour mieux expliquer.
Ce dernier a été réalisé à la suite des études accomplies dans les chapitres précédents.

Conclusion générale

Le travail que nous avons décrit dans ce mémoire avait pour objet la recherche des experts scientifiques en utilisant le *deep learning*, et notamment les modèles de langues préentrainés basés sur le modèle *BERT* proposé par *Google*. Nous nous sommes pour cela appuyés sur les travaux de Berger et al. [5]. Nous avons exploré trois pistes d'améliorations par rapport à ces travaux : 1) une nouvelle méthode d'indexation pour les documents, qui est l'indexation par phrases 2) utilisation de deux méthodes pour augmenter la requête, en extrayant la définition depuis *Wikipedia*, 3) Modification de la formule de vote en incorporant l'information du domaine de l'auteur. Nous avons combiné toutes ces méthodes avec deux variantes du modèle *BERT*, que sont *SciBERT* et *RoBERTa*, pour comparer avec nos propositions d'améliorations. Le constat est que notre modèle le plus performant a dépassé l'état de l'art sur la base de données, Arxiv+MAG utilisée par les auteurs de ses travaux. La meilleure combinaison obtenue est celle utilisant l'indexation par phrases, avec *SciBERT* comme modèle de plongement, et avec augmentation de la requête.

Nous avons refait l'évaluation sur une nouvelle base de données, que nous avons nous mêmes créée en utilisant le *web scraping* (extraction automatisé de contenu web avec un script) sur la bibliothèque ACM, et les résultats confirment globalement que les méthodes que nous avons proposé sont meilleures que le modèle de base. Un autre résultat intéressant est que l'utilisation de la définition pour augmenter la requête a été très bénéfique. La modification que nous avons apporté à la formule de vote, est dans un stade préliminaire, et nous n'avons constaté que de minimes améliorations, et dans certains cas aucune. Cette modification n'en constitue pas moins une piste prometteuse nécessitant une investigation plus approfondie. Finalement, et à notre surprise, *RoBERTa* fut dans la majorité des cas meilleur pour l'indexation que *SciBERT*. bien que *SciBERT* a montré de bons résultats sur de petites phrases. Les méthodes proposées dans le cadre de notre projet ont été intégrées dans une plateforme que nous avons développé pour la recherche d'experts académiques Algériens. La tâche de recherche d'experts étant une tâche récurrente, dont l'automatisation donnerait beaucoup d'avantages, nous espérons à travers notre travail contribuer à une meilleure valorisation du milieu académique Algérien.

Pour finir, les résultats de nos recherches, sont prometteuses et peuvent être améliorées en considérant les perspectives suivantes : 1) Utilisation d'autres mots proches de la requête (par des métaheuristique par exemple) en plus de sa définition, nous pourrions également considérer l'utilisation du plongement par *SciBERT* lors de la détermination des mots proches de la requêtes, 2) Utilisation plus précise du domaine de l'auteur en considérant le cas du *clustering*, par exemple, 3) Amplification de la distance entre le domaine de l'auteur et du document pour rendre les changements plus significatifs, et obtenir ainsi une plus grande différence de scores

Annexe A

Complément de la réalisation et l'évaluation des approches proposées

A.1 Collecte et nettoyage des données

L'algorithme suivant, "**Word correction**" récapitule les étapes de corrections d'un mot :

Algorithm 4 Word correction

Require: word, wordAppears
Ensure: word
  $wordAppears = BSTSearch(word)$
  if wordAppears is False **then**
    $wordAppears = APIDictionnaireSearch(word)$
    if wordAppears is False **then**
      $word = ZipfLaw(word)$
    end if
  end if

L'algorithme suivant, **Data cleaning** résume les étapes de nettoyage des résumés :

Algorithm 5 Data cleansing

Require: abstract
Ensure: sentencesList
  $abstract = DeleteEmailAddressesURLs(abstract)$
  $abstract = EliminatePunctuation(abstract)$
  $abstract = RemoveExtraSpaces(abstract)$
  $abstract = TransformNumbersIntoWords(abstract)$
  $sentencesList = SplitIntoSentencesAtEachFullstop(abstract)$
  for each *sentenceinsentencesList* **do**
    for each *wordinsentence* **do**
      $WordCorrection(word)$
    end for
  end for

A.2 Les requêtes qui ont été utilisées dans l'évaluation des deux bases de données

A.2.1 Requêtes utilisées pour la base de données Arxiv+MAG

{'cluster analysis','Image segmentation','Parallel algorithm','Monte Carlo method','Convex optimization','Dimensionality reduction','Facial recognition system','k-nearest neighbors algorithm','Hierarchical clustering','Automatic summarization','Dynamic programming','Genetic algorithm','Human-computer interaction','Categorial grammar','Semantic Web','fuzzy logic','image restoration','generative model','search algorithm','sample size determination','anomaly detection','sentiment analysis','semantic similarity','world wide web','gibbs sampling','user interface','belief propagation','interpolation','wavelet transform','transfer of learning','topic model','clustering high-dimensional data','game theory','biometrics','constraint satisfaction','combinatorial optimization','speech processing','multi-agent system','mean field theory','social network','lattice model','automatic image annotation','computational geometry','Evolutionary algorithm','web search query','eye tracking','query optimization','logic programming','Hyperspectral imaging','Bayesian statistics','kernel density estimation','learning to rank','relational database','activity recognition','wearable computer','big data','ensemble learning','wordnet','medical imaging','deconvolution','Latent Dirichlet allocation','Euclidean distance','web service','multi-task learning','Linear separability','OWL-S','Wireless sensor network','Semantic role labeling','Continuous-time Markov chain','Open Knowledge Base Connectivity','Propagation of uncertainty','Fast Fourier transform','Security token','Novelty detection','semantic grid','Knowledge extraction','Computational biology','Web 2.0','Network theory','Video denoising','Quantum information science','Color quantization','social web','entity linking','information privacy','random forest','cloud computing','Knapsack problem','Linear algebra','batch processing','rule induction','Uncertainty quantification','Computer architecture','Best-first search','Gaussian random field','Support vector machine','ontology language','machine translation','middleware',"Newton's method"}

A.2.2 Requêtes utilisées pour la base de données ACM

{'machine learning','information retrieval','neural networks','data mining','logic','design and analysis of algorithms','computer vision','cluster analysis','mathematical optimization','natural language processing','image manipulation','feature selection','graph algorithms','artificial intelligence','enterprise computing','systems biology','object recognition','deep learning','knowledge representation and reasoning','genetics','discrete mathematics','video segmentation','sequential decision making','graph theory','human centered computing','computational geometry','probabilistic reasoning','robotics','dynamic programming','web applications','network protocols','evaluation','animation',' recognition','information theory','motion capture','computational biology','markov decision processes','models of computation','image processing','computer graphics','cognitive science','numerical analysis','computer crime','image segmentation','video summarization','scheduling algorithms','coding theory','modeling and simulation','social media','program semantics','personalization','in systems','active learning','economics','sound and music computing','computer supported cooperative work','health informatics','image compression','electronic commerce','model checking','geographic visualization','big data','cognitive robotics','augmented reality','parallel algorithms','planning and scheduling','question answering','dimensionality reduction','ontologies','activity recognition','support vector machines','human robot interaction','natural language generation','human computer interaction','xml','combinatorial optimization','network security','privacy policies','online learning','data analytics','scalability','memory management','evolutionary computation','information integration','internet of things','algorithmic mechanism design','topic modeling','people with disabilities','gpgpu','logic programming','concept drift','data warehouses','systems analysis and

design','conditional random fields','web conferencing','similarity search','distributed algorithm','autonomous vehicles','adaptive filtering'}

A.3 L'architecture des bases de données utilisées

A.3.1 Base de données Arxiv + MAG

Base de données	Attribut	Type	Désignation
Auteurs	id	N	L'identifiant de l'auteur
	n_citation	N	Nombre de citations de l'auteur (la somme des citations de tous ses articles)
	n_pubs	N	Nombre de publications de l'auteur
	name	A	Nom d'auteur
	pubs	AN	Une liste des identifiants de ses articles
	tags	AN	Le domaine de travail (ensemble de termes) présent dans l'ensemble de données MAG pour les auteurs en tant que proxy pour évaluer la pertinence d'un auteur
Documents	id	N	L'identifiant du document
	title	AN	Titre du document
	authors	AN	Informations sur les auteurs du document : nom, identifiant et organisme
	venue	AN	La revue du document
	year	N	Année de publication
	n_citation	N	Nombre de citations
	page_start	N	Numéro de la première page
	page_end	N	Numéro de la dernière page
	doc_type	A	Type de document : journal, conférence, livre
	publisher	AN	Éditeur
	volume	N	Volume
	issue	N	édition
	fos	AN	Le domaine de travail
	doi	AN	Identificateur d'objet numérique
	references	N	Références
	abstract	AN	Résumé
	cleaned_abstract_sentences	AN	Phrases nettoyées du résumé

TABLE A.1 – L'architecture de la base de données Arxiv+MAG

A.3.2 Base de données ACM (International)

Base de données	Attribut	Type	Désignation
Auteurs	author_id	N	L'identifiant de l'auteur
	author_name	A	Nom d'auteur
	author_average_citation_per_article	N	Citation moyenne par article
	author_citation_count	N	Nombre de citations de l'auteur (la somme des citations de tous ses articles)
	author_publication_counts	N	Nombre de publications de l'auteur
	papers_available_for_download	N	Nombre d'articles disponibles pour le téléchargement
	pubs	AN	Une liste des identifiants de ses articles
	tags	AN	Domaines d'activité de l'auteur
Documents	id_paper	AN	L'identifiant du document
	title	AN	Titre du document
	abstract	AN	Résumé
	cleaned_abstract_sentences	A	Phrases nettoyées du résumé
	paper_citation	N	Nombre de citations
	revue	AN	Détails de l'édition [revue, volume, édition, date, pages, URL de l'article]
	authors	AN	Liste des auteurs qui ont rédigé le document : nom et identifiant

TABLE A.2 – L'architecture de la base de données ACM (International)

A.3.3 Base de données OpenAlex + ACM (Algérienne)

Base de données	Attribut	Type	Désignation
Auteurs	id	N	L'identifiant de l'auteur
	name	A	Nom d'auteur
	n_pub	N	Nombre de publications de l'auteur
	tags	AN	Domaines d'activité de l'auteur
	papers	N	Une liste des identifiants de ses articles
	author_institution	AN	Institution de l'auteur
Documents	id_paper	AN	L'identifiant du document
	title	AN	Titre du document
	abstract	AN	Résumé
	cleaned_abstract_sentences	A	Phrases nettoyées du résumé
	authors	N	Liste des auteurs qui ont rédigé le document (identifiant)

TABLE A.3 – L'architecture de la base de données OpenAlex + ACM (Algérienne) utilisé dans le site web

Bibliographie

[1] Dominik Kuropka. *Modelle zur Repräsentation natürlichsprachlicher Dokumente*. Logos-Verl., 2004.

[2] Bhaskar Mitra, Nick Craswell, et al. An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval*, 13(1) :1–126, 2018.

[3] APS. Conseil interministériel consacré à la promotion de la recherche et du développement dans les entreprises. `https://www.aps.dz/economie/96376`, 27-10-2019.

[4] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions : a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7 :453–466, 2019.

[5] Mark Berger, Jakub Zavrel, and Paul Groth. Effective distributed representations for academic expert search. *arXiv preprint arXiv :2010.08269*, 2020.

[6] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert : A pretrained language model for scientific text. *arXiv preprint arXiv :1903.10676*, 2019.

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta : A robustly optimized bert pretraining approach. *arXiv preprint arXiv :1907.11692*, 2019.

[8] CERIST. A propos du cerist-historique. `http://www.cerist.dz/index.php/fr/apropsducerist-2/176-historique`, 13-06-2022.

[9] CERIST. A propos du cerist-missions. `http://www.cerist.dz/index.php/fr/apropsducerist-2/173-missions`, 13-06-2022.

[10] DTISI. Qui sommes-nous ? `http://dtisi.cerist.dz/index.php/2019/02/05/qui-sommes-nous/`, 13-06-2022.

[11] CERIST. Sécurité informatique. `http://www.cerist.dz/index.php/fr/rechercheetdevelop/114-divisions-de-recherche/913-securite-informatique`, 13-06-2022.

[12] CERIST. Systèmes d'information et systèmes multimédia. `http://www.cerist.dz/index.php/fr/rechercheetdevelop/114-divisions-de-recherche/524-systemes-d-information-et-systemes-multimedia-fr`, 13-06-2022.

[13] CERIST. Réseaux et systèmes distribués. `http://www.cerist.dz/index.php/fr/rechercheetdevelop/114-divisions-de-recherche/523-recherche-et-developpement-en-reseaux-fr`, 13-06-2022.

[14] DRDHN. Qui sommes-nous ? `http://www.drdhn.cerist.dz/?page_id=18`, 13-06-2022.

[15] Stefan Buttcher, Charles LA Clarke, and Gordon V Cormack. *Information retrieval : Implementing and evaluating search engines*. Mit Press, 2016.

[16] Hinrich Schutze Christopher D. Manning, Prabhakar Raghavan. Introduction to information retrieval. *Cambridge University Press*, 2008.

[17] Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. A dual embedding space model for document ranking. *arXiv preprint arXiv :1602.01137*, 2016.

[18] Amit Singhal et al. Modern information retrieval : A brief overview. *IEEE Data Eng. Bull.*, 24(4) :35–43, 2001.

[19] Gabriella Pasi. Fuzzy sets in information retrieval : State of the art and research trends. *Fuzzy Sets and Their Extensions : Representation, Aggregation and Models*, pages 517–535, 2008.

[20] Kenneth Kunen. Set theory, volume 34 of studies in logic, 2011.

[21] FW Lancaster and EG Fayen. Information retrieval on-line, melville publ. *Co., Los Angeles, Calif*, page 15, 1973.

[22] Stephen E Robertson and K Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3) :129–146, 1976.

[23] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11) :613–620, 1975.

[24] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11) :964–971, 1987.

[25] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6) :391–407, 1990.

[26] Robert Price, Anthony Zukas, et al. Application of latent semantic indexing to processing of noisy text. In *International Conference on Intelligence and Security Informatics*, pages 602–603. Springer, 2005.

[27] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. In *Linear algebra*, pages 134–151. Springer, 1971.

[28] Madeleine Udell, Corinne Horn, Reza Zadeh, Stephen Boyd, et al. Generalized low rank models. *Foundations and Trends® in Machine Learning*, 9(1) :1–118, 2016.

[29] Maria Fasoi, John Pavlopoulos, and Maria Konstantinidou. Computational authorship analysis of homeric language. In *Digital Humanities Workshop*, pages 78–88, 2021.

[30] Manish Gupta, Michael Bendersky, et al. Information retrieval with verbose queries. *Foundations and Trends® in Information Retrieval*, 9(3-4) :209–354, 2015.

[31] Djoerd Hiemstra and Franciska de Jong. Disambiguation strategies for cross-language information retrieval. In *International Conference on Theory and Practice of Digital Libraries*, pages 274–293. Springer, 1999.

[32] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Frederick Jelinek, John Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2) :79–85, 1990.

[33] David JC MacKay and Linda C Bauman Peto. A hierarchical dirichlet language model. *Natural language engineering*, 1(3) :289–308, 1995.

[34] Olga Vechtomova. Query expansion for information retrieval., 2009.

[35] Amanda Spink, Dietmar Wolfram, Major BJ Jansen, and Tefko Saracevic. Searching the web : The public and their queries. *Journal of the American society for information science and technology*, 52(3) :226–234, 2001.

[36] Statista. Average number of search terms for online search queries in the united states as of august 2017.

[37] Hiteshwar Kumar Azad and Akshay Deepak. Query expansion techniques for information retrieval : a survey. *Information Processing & Management*, 56(5) :1698–1735, 2019.

[38] Melvin Earl Maron and John Larry Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM (JACM)*, 7(3) :216–244, 1960.

[39] Saar Kuzi, Anna Shtok, and Oren Kurland. Query expansion using word embeddings. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 1929–1932, 2016.

[40] Ilyes Khennak and Habiba Drias. Bat-inspired algorithm based query expansion for medical web information retrieval. *Journal of medical systems*, 41(2) :1–16, 2017.

[41] Ling Liu and M Tamer Özsu. *Encyclopedia of database systems*, volume 6. Springer, 2009.

[42] Shahrzad Naseri, Jeffrey Dalton, Andrew Yates, and James Allan. Ceqe : Contextualized embeddings for query expansion. In *European Conference on Information Retrieval*, pages 467–482. Springer, 021.

[43] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3) :225–331, 2009.

[44] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th international conference on world wide web*, pages 1291–1299, 2017.

[45] Anders Brahme. *Comprehensive biomedical physics*. Newnes, 2014.

[46] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biology*, 52(1), 1990.

[47] Weihong Zhang and Ying Zhou. *The Feature-Driven Method for Structural Optimization*. Elsevier, 2020.

[48] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[49] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1) :1–127, 2009.

[50] Jürgen Schmidhuber. Deep learning in neural networks : An overview. *Neural networks*, 61 :85–117, 2015.

[51] Kalyan Das, Jiming Jiang, and JNK Rao. Mean squared error of empirical predictor. *The Annals of Statistics*, 32(2) :818–840, 2004.

[52] Yaoshiang Ho and Samuel Wookey. The real-world-weight cross-entropy loss function : Modeling the costs of mislabeling. *IEEE Access*, 8 :4806–4813, 2019.

[53] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31, 2018.

[54] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5) :185–196, 1993.

[55] Maria V Valueva, NN Nagornov, Pavel A Lyakhov, Georgii V Valuev, and Nikolay I Chervyakov. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and computers in simulation*, 177 :232–243, 2020.

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[57] Zellig S Harris. Distributional structure. word 10, 2-3 (1954), 146–162. *Google Scholar Google Scholar Cross Ref Cross Ref*, 1954.

[58] JR Firth. Papers in linguistics 1934–1951, london 1957. *A Synopsis of Linguistic Theory*, 1955 :1–32, 1930.

[59] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[60] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*, 2013.

[61] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv :1810.04805*, 2018.

[62] CERIST. Systèmes d'information et systèmes multimédia. `https://www.oxfordlearnersdictionaries.com/definition/english/expert_1`, 13-06-2022.

[63] Mahmood Neshati, Zohreh Fallahnejad, and Hamid Beigy. On dynamicity of expert finding in community question answering. *Information Processing & Management*, 53(5) :1026–1042, 2017.

[64] Duen-Ren Liu, Yu-Hsuan Chen, Wei-Chen Kao, and Hsiu-Wen Wang. Integrating expert profile, reputation and link analysis for expert finding in question-answering websites. *Information processing & management*, 49(1) :312–329, 2013.

[65] G Alan Wang, Jian Jiao, Alan S Abrahams, Weiguo Fan, and Zhongju Zhang. Expertrank : A topic-aware expert finding algorithm for online knowledge communities. *Decision support systems*, 54(3) :1442–1451, 2013.

[66] Jianshan Sun, Wei Xu, Jian Ma, and Jiasen Sun. Leverage raf to find domain experts on research social network services : A big data analytics methodology with mapreduce framework. *International Journal of Production Economics*, 165 :185–193, 2015.

[67] Qi Wang, Jian Ma, Xiuwu Liao, and Wei Du. A context-aware researcher recommendation system for university-industry collaboration on r&d projects. *Decision Support Systems*, 103 :46–57, 2017.

[68] Yi Fang, Luo Si, and Aditya Mathur. Facfinder : Search for expertise in academic institutions. *Department of Computer Science, Purdue University, Tech. Rep. SERC-TR-294*, 2008.

[69] Yi Fang, Luo Si, and Aditya Mathur. Facfinder : Search for expertise in academic institutions. *Department of Computer Science, Purdue University, Tech. Rep. SERC-TR-294*, 2008.

[70] Omayma Husain, Naomie Salim, Rose Alinda Alias, Samah Abdelsalam, and Alzubair Hassan. Expert finding systems : A systematic review. *Applied Sciences*, 9(20) :4250, 2019.

[71] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. A language modeling framework for expert finding. *Information Processing & Management*, 45(1) :1–19, 2009.

[72] Ming Li, Lu Liu, and Chuan-Bo Li. An approach to expert recommendation based on fuzzy linguistic method and fuzzy text classification in knowledge management systems. *Expert Systems with Applications*, 38(7) :8586–8596, 2011.

[73] Cem Tekin, Onur Atan, and Mihaela Van Der Schaar. Discover the expert : Context-adaptive expert selection for medical diagnosis. *IEEE transactions on emerging topics in computing*, 3(2) :220–234, 2014.

[74] Wu Chen, Wei Lu, and Shuguang Han. Designing and implementation of expertise search & hotspot detecting system. In *INC2010 : 6th International Conference on Networked Computing*, pages 1–5. IEEE, 2010.

[75] Meredith Taylor and Debbie Richards. Finding and validating expertise. 2011.

[76] Jianshan Sun, Wei Xu, Jian Ma, and Jiasen Sun. Leverage raf to find domain experts on research social network services : A big data analytics methodology with mapreduce framework. *International Journal of Production Economics*, 165 :185–193, 2015.

[77] Balázs Sziklai. How to identify experts in a community ? *International Journal of Game Theory*, 47(1) :155–173, 2018.

[78] Muhammad Faisal, Ali Daud, and Abubakr Akram. Expert ranking using reputation and answer quality of co-existing users. *International Arab Journal of Information Technology (IAJIT)*, 14(1), 2017.

[79] Chaoran Huang, Lina Yao, Xianzhi Wang, Boualem Benatallah, and Quan Z Sheng. Expert as a service : Software expert recommendation via knowledge domain embeddings in stack overflow. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 317–324. IEEE, 2017.

[80] Majid Rafiei and Ahmad A Kardan. A novel method for expert finding in online communities based on concept map and pagerank. *Human-centric computing and information sciences*, 5(1) :1–18, 2015.

[81] Muhammad Tanvir Afzal and Hermann A Maurer. Expertise recommender system for scientific community. *J. Univers. Comput. Sci.*, 17(11) :1529–1549, 2011.

[82] Meredith Taylor and Debbie Richards. Finding and validating expertise. 2011.

[83] TREC. Text retrieval conferences. `https://trec.nist.gov/`, 22-07-2022.

[84] Christos Faloutsos and Douglas W Oard. A survey of information retrieval and filtering methods. Technical report, 1998.

[85] Y Connie Yuan, Janet Fulk, Peter R Monge, and Noshir Contractor. Expertise directory development, shared task interdependence, and strength of communication network ties as multilevel predictors of expertise exchange in transactive memory work groups. *Communication Research*, 37(1) :20–47, 2010.

[86] Rob Cross and Stephen P Borgatti. *The ties that share : Relational characteristics that facilitate information seeking*. na, 2004.

[87] Duen-Ren Liu, Yu-Hsuan Chen, Wei-Chen Kao, and Hsiu-Wen Wang. Integrating expert profile, reputation and link analysis for expert finding in question-answering websites. *Information processing & management*, 49(1) :312–329, 2013.

[88] Elena Smirnova and Krisztian Balog. A user-oriented model for expert finding. In *European Conference on Information Retrieval*, pages 580–592. Springer, 2011.

[89] Sharoda A Paul. Find an expert : Designing expert selection interfaces for formal help-giving. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 3038–3048, 2016.

[90] Shuyi Lin, Wenxing Hong, Dingding Wang, and Tao Li. A survey on expert finding techniques. *Journal of Intelligent Information Systems*, 49(2) :255–279, 2017.

[91] Guoping Hu, Jingjing Liu, Hang Li, Yunbo Cao, Jian-Yun Nie, and Jianfeng Gao. A supervised learning approach to entity search. In *Asia Information Retrieval Symposium*, pages 54–66. Springer, 2006.

[92] Krisztian Balog, Yi Fang, Maarten De Rijke, Pavel Serdyukov, Luo Si, et al. Expertise retrieval. *Foundations and Trends® in Information Retrieval*, 6(2–3) :127–256, 2012.

[93] Yunbo Cao, Jingjing Liu, Shenghua Bao, and Hang Li. Research on expert search at enterprise track of trec 2005. In *TREC*, 2005.

[94] Craig Macdonald. *The voting model for people search*. PhD thesis, University of Glasgow, 2009.

[95] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan) :993–1022, 2003.

[96] Xiaoyong Liu and W Bruce Croft. Cluster-based retrieval using language models. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, 2004.

[97] Hongbo Deng, Irwin King, and Michael R Lyu. Formal models for expert finding on dblp bibliography data. In *2008 Eighth IEEE International Conference on Data Mining*, pages 163–172. IEEE, 2008.

[98] Aidan Hogan and Andreas Harth. The expertfinder corpus 2007 for the benchmarking and development of expertfinding systems. 2007.

[99] Philipp Sorg, Philipp Cimiano, Antje Schultz, and Sergej Sizov. Overview of the cross-lingual expert search (cries) pilot challenge. In *CLEF (Notebook Papers/LABs/Workshops)*. Citeseer, 2010.

[100] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer : extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998, 2008.

[101] ARXIV. Arxiv homepage. `https://arxiv.org/`, 16-06-2022.

[102] Krisztian Balog, Yi Fang, Maarten De Rijke, Pavel Serdyukov, Luo Si, et al. Expertise retrieval. *Foundations and Trends® in Information Retrieval*, 6(2–3) :127–256, 2012.

[103] Meta AI. tools/faiss. `https://ai.facebook.com/tools/faiss/`, 07-06-2022.

[104] Craig Macdonald and Iadh Ounis. Voting techniques for expert search. *Knowledge and information systems*, 16(3) :259–280, 2008.

[105] Edilson A Corrêa Jr, Filipi N Silva, Luciano da F Costa, and Diego R Amancio. Patterns of authors contribution in scientific manuscripts. *Journal of Informetrics*, 11(2) :498–510, 2017.

[106] Python Software Foundation. Wikipedia-api. `https://pypi.org/project/Wikipedia-API/`, 16-06-2022.

[107] W3C. Python introduction. https ://www.w3schools.com/python/python$_{i}ntro.asp, 01 - 09 - 2022$.

[108] Anaconda. products/distribution. `https://www.anaconda.com/products/distribution`, 07-06-2022.

[109] Spyder. page d'accueil. `https://www.spyder-ide.org/`, 07-06-2022.

[110] Nvidia. Cuda zone. https ://developer.nvidia.com/cuda-zone, 01-09-2022.

[111] PyPI. project/sentence-transformers. `https://pypi.org/project/sentence-transformers/`, 07-06-2022.

[112] Scikit-learn. Getting started. https ://scikit-learn.org/stable/getting$_{s}tarted.html, 01 - 09 - 2022$.

[113] PyPI. project/requests. `https://pypi.org/project/requests/`, 07-06-2022.

[114] Python Software Foundation. Wikipedia api. https ://pypi.org/project/Wikipedia-API/, 01-09-2022.

[115] arXiv. A propos d'arxiv. https://arxiv.org/about, 21-08-2022.

[116] arXiv. Page d'accueil d'arxiv. https://arxiv.org/, 21-08-2022.

[117] Microsoft. Page de microsoft academic graph. https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/, 21-08-2022.

[118] ACM Digital Library. A propos d'acm digital library. https://dl.acm.org/about, 21-08-2022.

[119] Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi. Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST*, volume 4, page 1, 2012.

[120] Daniel Glez-Peña, Anália Lourenço, Hugo López-Fernández, Miguel Reboiro-Jato, and Florentino Fdez-Riverola. Web scraping technologies in an api world. *Briefings in bioinformatics*, 15(5) :788–797, 2014.

[121] Jason Brownlee. *Data preparation for machine learning : data cleaning, feature selection, and data transforms in Python*. Machine Learning Mastery, 2020.

[122] Micheline Petruszewycz. L'histoire de la loi d'estoup-zipf : documents. *Mathématiques et sciences humaines*, 44 :41–56, 1973.

[123] Suyash458. Wiktionaryparser. https ://github.com/Suyash458/WiktionaryParser, 03-09-2022.

[124] Miquel Porta. *A dictionary of epidemiology*. Oxford university press, 2008.

[125] python. random — generate pseudo-random numbers - functions-for-sequences. https://docs.python.org/3/library/random.html#functions-for-sequences, 23-08-2022.

[126] W3C. Html introduction. https://www.w3schools.com/html/html_intro.asp, 01-09-2022.

[127] W3C. Css introduction. https://www.w3schools.com/css/css_intro.asp, 01-09-2022.

[128] Futura Tech. Javascript : qu'est-ce que c'est ? https://www.futura-sciences.com/tech/definitions/internet-javascript-509/, 01-09-2022.

[129] Pallets. Flask. https ://palletsprojects.com/p/flask/, 01-09-2022.

[130] Pallets. Jinja. https://jinja.palletsprojects.com/en/3.1.x/, 01-09-2022.