الجـــــــــمهـــــورية الجـــــــــزائريـــة الديمقـــــــــراطيـــــــة الشـــــــــعبيـــــة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليـــــــــــــــــــــم العـــــــــــالي والبحـــث العـــــــــــلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــــــامعة غـــــردايـــــــة

Université de Ghardaia

كـــــــليـــــة العلـــــــوم والتكنـــــولوجيـــــــا

Faculté des Sciences et de Technologie

قسم الرياضيات والإعلام الآلي

Département des Mathématiques et Informatique

Laboratoire de Mathématiques et Sciences Appliquées

مخبر الرياضيات و العلوم التطبيقية

Laboratory of Mathematics and Applied Sciences

MÉMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En : Informatique

Spécialité : Systèmes Intelligents pour l'Extraction de Connaissances

Par : Aissa ZENDARI

Thème

Algorithme des abeilles pour la recherche d'information

Soutenu publiquement le 03/07/2019 devant le jury composé de :

M. Slimane BELLAOUAR	MCB	Univ. Ghardaia	Président
M. Khaled KECHIDA	MAA	Univ. Ghardaia	Examinateur
M. Abdelkader OULD MAHREZ	MAA	Univ. Ghardaia	Examinateur
M. Abdelkader BOUHANI	MAA	Univ. Ghardaia	Directeur de mémoire

Année Universitaire 2018/2019

Dédicaces

Je dédie ce modeste travail à

Mon très cher père *qui m'a inconditionnellement soutenu non seulement durant mon parcours universitaire mais depuis toujours et veillé à ce que j'aie toutes les conditions favorables pour étudier.*

Ma très chère mère *qui m'a entouré d'une tendresse et d'une compréhension favorables pour l'apprentissage qui a veillé à mon éducation et qui quoi que je fasse je ne pourrai pas lui rendre ne serai est ce qu'une infime partie de ce qu'elle m'a donné.*

Mon épouse bien aimée*, qui ne cesse de me motiver, et qui veille à m'entourer de confort et de soutien*

Ma chère sœur *qui a été la sœur et l'amie à la fois et qui en sa compagnie je passais des moments inoubliables.*

Mes deux enfants *qui créent de l'ambiance de la joie et de la gaieté partout où ils passent.*

Ma chère grand-mère *dont les conseils et la raison me guident lorsque je perds les repères.*

Mon oncle *avec qui j'ai appris pleins de choses et envers qui je suis très reconnaissant pour son amitié et sa compréhension.*

Mes tantes *qui me réservent un amour inconditionné.*

A la mémoire de ***ma grand-mère*** *et de mes* ***grands-pères****.*

A mes neveux et nièces.

A mes cousins et cousines.

Aux amis du primaire qui occupent une place particulière.

Aux amis depuis le lycée qui sont plus des frères que des amis.

A toutes les personnes que je connais et que je n'ai pas citées.

Aissa

Remerciements

Je tiens avant tout à remercier dieu de m'avoir donné le courage la force et la volonté pour achever ce travail.

Je remercie spécialement ***M. Yacine HADJMOUSSA*** *pour sa serviabilité, sa gentillesse et sa générosité.*

Je tiens à remercier très spécialement ***M.Abdelkader BOUHANI*** *d'abord pour m'avoir introduit aux systèmes multi-agent et surtout pour son suivi et ses conseils précieux dans la direction de ce travail.*

Je remercie infiniment tous le cadre enseignant de la spécialité SIEC à l'université de ***GHARDAIA,*** *et spécialement ceux m'ayant enseigné cette année, les chers et respectables messieurs :* ***Dr.BELLAOUAR Slimane****,* ***Dr.KERRACHE Chaker Abdelaziz****,* ***M.MAHDJOUB Youcef*** *et* ***Prof.MOUSSAOUI Abdelouahab*** *pour leur enseignements leur indulgence et leur dévouement.*

Je remercie mes collègues à l'université de ***GHARDAIA.***

Je tiens à remercier le cadre administratif du département MI et de la faculté des Sciences et de la Technologie.

Je tiens aussi à remercier et à marquer ma reconnaissance à tous les enseignants de l'ESI spécialement ceux d'Organisation de MCSI de Math et d'Analyse de données pour leur dévouement et abnégation dans notre instruction au sein de cette école.

A tous les enseignants depuis le primaire qui ont fait de moi ce que je suis.

Enfin je tiens à remercier toutes les personnes qui ont contribué de près ou de loin à l'accomplissement de ce travail.

ملخص

تتناول هذه الأطروحة توفير حاجات المستخدم من المعلومات عند استعماله أنظمة استرجاع المعلومات التي تستخدم مجموعات من المستندات كبيرة الحجم وهذا خلال وقت استجابة معقول؛ وهذا باستخدام خوارزمية حديثة وفعالة وهي خوارزمية مستعمرة النحل الاصطناعية (ABC) تحاكي هذه الخوارزمية السلوك الذكي والتعاوني للنحل في البحث عن أماكن غنية بالغذاء.لقد تم لهذا الغرض انشاء برنامج يقوم أولا بتهيئة المستندات وذلك عن طريق التجميع وكذلك نظام بحث عن طريق مستعمرة النحل الاصطناعية بغية استكشاف فضاء البحث وتقليل الوقت اللازم للإستجابة . تمت برمجة محرك بحث يدمج النظاميين المذكورين أعلاه أي برنامج التجميع وبرنامج البحث ويسمح بإجراء التجارب المختلفة ، مقارنتها وجرد نتائجها .التجارب تمت باستخدام مجموعة مستندات مجلد رويترز1 (RCV1) ، وتظهر فعالية الطريقة المقترحة فيما يخص وقت الاستجابة بالمقابل تظل نوعية المستندات المسترجعة مرهونة بنوعية التهيئة المطبقة.

كلمات مفتاحية :

تحسين ، علم استرجاع المعلومات ، مجموعات مستندات كبيرة ، وقت الاستجابة ، مستعمرة النحل الاصطناعية ،فضاء البحث .

Résumé

Le sujet abordé par ce mémoire est la sélection des documents répondant aux besoins de l'utilisateur dans les systèmes de recherche d'information dans les larges collections de documents dans un délai raisonable et ce avec une approche moderne et robuste basée sur l'Algorithme des colonies d'abeilles artificielles (Artificial Bee Colony Algorithm). Cette approche comme son nom l'indique s'inspire du comportement intelligent et collaboratif des abeilles fourragères lors de la recherche de sites de nourriture abondants. Un prétraitement de la collection à l'aide d'une segmentation et un processus de recherche par les abeilles sont conçus afin d'explorer efficacement l'espace de recherche et améliorer en conséquence le temps de réponse. Ainsi un moteur de recherche concrétisant les approches proposées ainsi qu'une approche classique afin de comparer entre les deux et relever les résultats, a été réalisé. Les tests ont été menés sur la collection de documents : Reuters Corpus volume 1 (RCV1) et montrent l'efficacité de l'approche proposée en terme de temps de réponse. Par contre la qualité des documents retournés reste très dépendante de celle du prétraitement effectué.

Mots clés : recherche d'information, larges collections, temps de réponse, colonies d'abeilles artificielles, espace de recherche.

Abstract

This thesis deals with satisfying user's needs while using information retrieval systems that use large-scale document collections insuring that this retrieval occurs in a reasonable time ; it addresses this issue using a powerful and modern approach, which is the artificial bee colony optimization (ABC). This approach mimics the intelligent yet collaborative behavior of foraging honeybees in their quest for rich food sources. A pretreatment of the document collection using clustering and an ABC based search process are designed to efficiently explore the search space and thus minimize the response time. In addition, a search engine is produced, it includes the proposed approaches described above and a classical approach in order to makes tests between both classical and proposed approaches and to compare results of the bees' algorithm regarding its different parameters. The tests are conducted on the Reuters Corpus volume 1 (RCV1) and they show the efficiency of the proposed approach on time response. For the quality of the retrieved documents, it is largely relying on the quality of the pretreatment performed before the search process**.**

Keywords: Information retrieval, large-scale collections, response time, artificial bee colony,search space.

Table des matières

Chapitre 02 : État de l'art

Chapitre 03 : Conception et implémentation de l'approche proposée

Table des figures

Liste des tableaux

Table des abréviations

Abréviation	Signification
RI	Recherche d'information
SRI	Système de recherche d'information
SGBD	Système de Gestion de bases de données
ABC	Colonie d'abeilles artificielles
BSO	Optimisation par les essaims d'abeilles
AA	Apprentissage Automatique
IA	Intelligence artificielle

Introduction générale

Selon le cabinet d'analystes IDC (Internet Data Corporation), de novembre 2018 à 2025, le volume mondial de production de données devrait être multiplié par plus de huit pour atteindre 163 Zetta-octets (soit 1 000 milliards de giga-octets) [W4]. Ce phénomène d'explosion exponentielle de la masse de données mondiale relève de nouveaux défis quant à la sécurité et à l'efficacité d'exploitation de ces volumes importants. Le domaine de recherche d'information est l'un des secteurs informatiques affectés par cette évolution effrénée, et certaines méthodes classiques de recherche d'information présentent des soucis d'efficacité face à des données d'une telle envergure, la collection de documents GOV2 à titre d'exemple contient 25 205 179 documents [4], d'où la nécessité de techniques novatrices plus robustes pour aborder ses problèmes de performance.

La présente étude s'intéresse à l'une de ces techniques : *"* l'optimisation par colonies d'abeilles artificielles *"*, qui est une méta-heuristique bio-inspirée robuste, et son application dans le domaine de la recherche d'information. Elle s'intéresse aussi à la possible combinaison de cette technique à d'autres techniques d'apprentissage automatique tel que le clustering afin de faciliter la convergence de l'algorithme des abeilles. Ce choix est Motivé par le succès de cette technique pour les cas d'optimisation numériques [10].

Cette étude est à la croisée de trois domaines informatiques à savoir : la recherche d'information, l'intelligence des essaims et l'apprentissage automatique.

Problématique :

La sélection des documents pertinents par rapport à une requête formulée par l'utilisateur est processus assez long quand il s'agit d'interroger des collections de documents très volumineuses par les méthodes classiques de recherche d'information ce qui nuit à son expérience utilisateur lors de la manipulation du système de recherche.

L'étude a pour objectif d'exploiter les mécanismes intelligents inspirés des abeilles et proposer une adaptation de l'algorithme des abeilles artificielles au problème de recherche d'information afin de retourner des documents pertinents par rapport à une requête dans un délai acceptable.

Phases de la recherche :

Afin de répondre à la problématique, il a été procédé comme suit :

- Collecte des informations et connaissances dans les 3 domaines informatiques ayant trait au sujet.
- Elaboration d'une étude bibliographique sur le sujet.
- Décortiquer l'algorithme des abeilles et l'apport qu'il nous procure dans la recherche d'information.

Plan du mémoire :

Ce mémoire est composé d'une introduction, de trois chapitres et une conclusion générale, qui sont organisés

Comme suit :

- **Le premier chapitre** regroupe les connaissances nécessaires pour cerner toutes les facettes du sujet tels que la recherche d'information et les concepts sous-jacents comme les modèles de recherche, le calcul de similarité...etc et aussi les concepts d'intelligence des essaims et l'algorithme des abeilles ainsi que l'apprentissage automatique et les concepts relatifs.
- **Le deuxième chapitre** constitue une étude de la littérature et des travaux réalisés dans le domaine du sujet étudié par la communauté scientifique avec des analyses et comparaisons de ces travaux.
- **Le troisième Chapitre** propose une approche pour la résolution du problème posé, montre l'implémentation de la solution, enregistre les expérimentations effectuées et discute les résultats obtenus.

La conclusion générale rappelle le sujet étudié et la problématique posée et donne une synthèse des contributions apportées.

Chapitre 01
Concepts théoriques

Introduction

Afin d'étudier la thématique abordée dans ce travail, des notions théoriques et des concepts d'intelligence artificielle sont nécessaires à connaitre afin de mieux comprendre la problématique posée et se munir des outils nécessaires pour y répondre de la plus cohérente et précise des manières.

Le sujet de cette thèse étant à l'intersection de trois disciplines d'intelligence artificielle en l'occurrence : La recherche d'information, l'intelligence des essaims et l'apprentissage automatique une étude dans ces domaines est intéressante, ainsi nous verrons la recherche d'information, ses modèles et les enjeux qu'elle présente. Nous verrons aussi l'intelligence des essaims et les grandes lignes de l'algorithme des colonies d'abeilles artificielles et enfin quelques notions et algorithmes d'apprentissage automatique.

1. Recherche d'information

1.1 Introduction à la recherche de l'information

Selon **[4]** La recherche d'information comprend des travaux sur un éventail de types d'information.

Historiquement l'attention a été portée d'abord sur les documents textuels. Les pages Web, les e-mails, les articles scientifiques, les livres et les articles de presse en sont que quelques exemples. Tous ces documents ont une certaine structure telle que le titre, la date, l'auteur etc.

Cependant la différence triviale entre un document et un enregistrement de bases de données est que la plupart de l'information du document est sous forme de texte qui est non structuré. Pour illustrer la différence en prend l'exemple de deux données le numéro de compte et le solde de celui-ci, les deux ont une structure bien définie (un entier à six chiffres pour le numéro de compte et un nombre réel avec deux chiffres après la virgule) dès lors il est facile de comparer deux valeurs ou faire des opérations arithmétiques. Prenons maintenant l'exemple d'articles de journaux, ceux-là ont bel et bien une certaine structure (Titre, Date etc.) mais le contenu principal est l'article lui-même. S'il on venait à stocker ces articles dans une base de données un large champ sera attribué au texte de l'article mais ce texte n'a pas de structure ou décomposition. Afin de répondre aux besoins en informations des utilisateurs on traite leurs requêtes qui sont souvent simpliste (quelques mots), parfois vagues et imprécises. Pour ce faire on aura besoins d'algorithmes capables de comprendre et décider si l'information souhaité par l'utilisateur est contenue dans ce champ ou dans le texte de l'article, or il est plus difficile de définir le sens d'une phrase ou d'un paragraphe que de manipuler des numéros de comptes et des soldes

La compréhension et la modélisation de la manière avec laquelle les gens comparent le texte, et la conception d'algorithmes qui font cette tâche est au cœur de la recherche d'information.

De plus en plus, les applications de recherche d'information impliquent des documents multimédias tels que les images la vidéo ou l'audio. Ces supports ont un contenu qui, comme le texte, est difficile à décrire et à comparer. Dans la recherche de documents non textuels on se référait aux descriptions textuelles de leur contenu plutôt que sur le contenu lui-même, mais actuellement des progrès sont en cours en ce qui concerne les techniques de comparaison directe d'images, par exemple.

La recherche d'information implique une gamme de tâches et d'applications. Le scénario de recherche habituel implique que quelqu'un tape une requête dans un moteur de recherche et reçoive des réponses sous la forme d'une liste de documents classés par ordre de mérite. Bien que la recherche sur le Web soit de loin l'application la plus courante impliquant la recherche d'information elle trouve d'importantes applications dans les entreprises. Celles-ci consistent à trouver les informations requises dans la grande variété de fichiers informatiques disséminés sur un intranet d'entreprise.

La recherche verticale est une forme spécialisée de recherche Web où le domaine de la recherche est limité à un sujet particulier. Il existe aussi une forme de recherche personnelle

appelée ‘’recherche Desktop ‘’ où les sources d'information sont les fichiers stockés sur un ordinateur individuel, y compris les messages électroniques et les pages Web récemment consultées

On peut ainsi définir la recherche d'information comme suit :

Définition1 :

‘’ *La recherche de l'information est un domaine qui s'intéresse à la représentation, l'analyse, l'organisation, le stockage, la recherche et la sélection de l'information*’’. **[4]**

Definition2 :

‘’ La recherche d'information (IR) consiste à trouver du matériel (généralement des documents) d'une nature non structurée (généralement du texte) qui répond à un besoin d'information à partir de grandes collections (généralement stockées sur des ordinateurs)’’ **[5]**

1.2 Architecture d'un système de recherche d'information

Selon **[6] l'architecture d'un SRI est comme suit :**

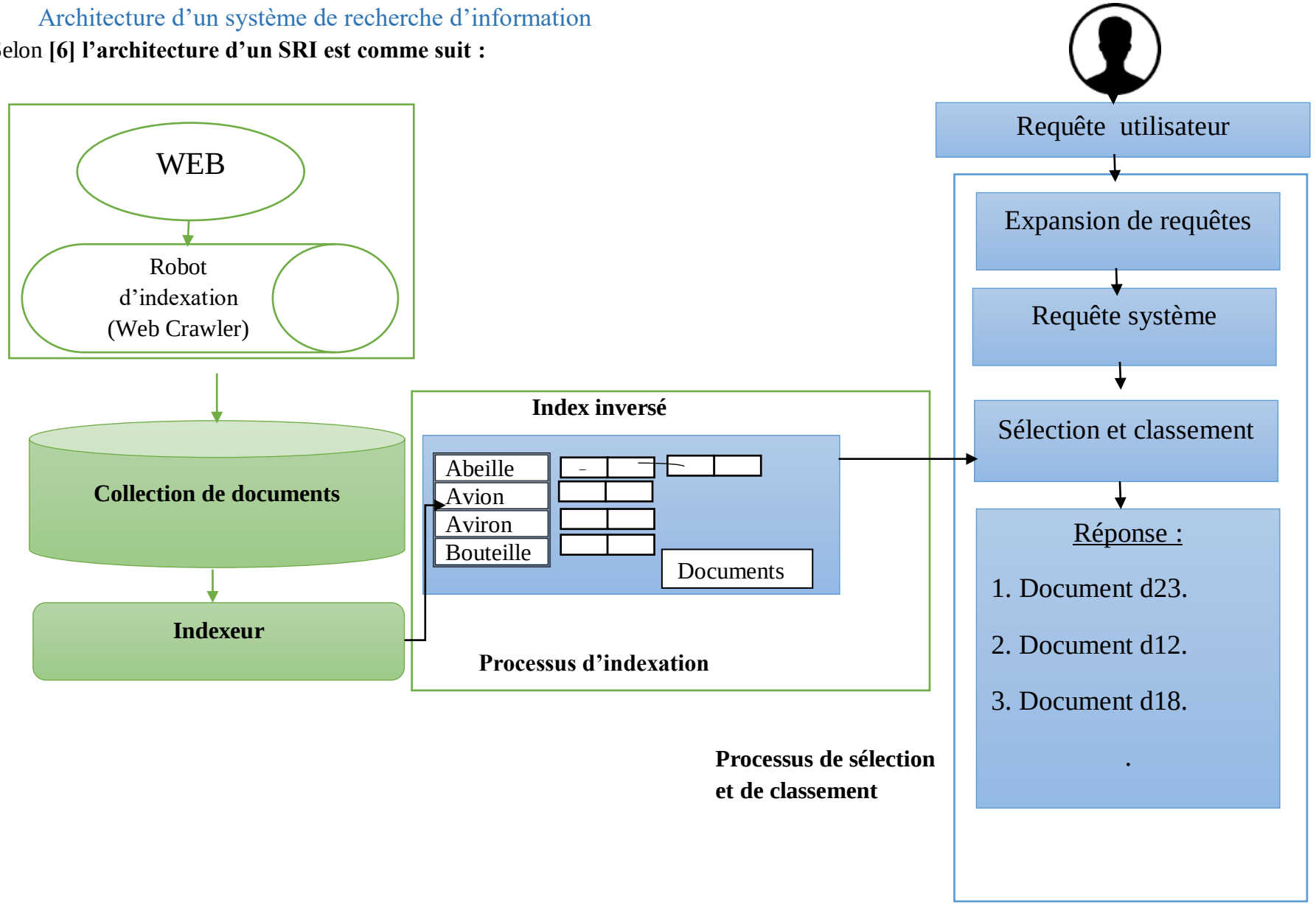


Figure 1: Architecture d'un système de recherché d'information

1.3 Représentation de l'information

L'indexation est le processus permettant de donner une représentation utile et pratique du contenu d'un document, elle assure l'efficacité et la performance lors de la recherche d'information.

L'index contient un ensemble de termes (mots clés), ces mots clés peuvent êtres dérivés automatiquement (indexation automatique) comme ils peuvent être réalisés par un expert (indexation manuelle), la combinaison des deux techniques s'appelle l'indexation semi-automatique. Les mots clés d'un index offrent une vue logique sur le document.

Pour des collections de documents réduite, il est possible de représenter un document par tout l'ensemble de ses mots, ce qui constituera une vue logique complète de celui-ci mais cette pratique engendre des coûts de calcul élevés **[6]**.

Avec de large collections par contre nous devons réduire l'ensemble de mots représentatifs pour un document. Cette tâche est accomplie à travers l'**élimination des mots** vides qui ne sont pas porteur de sens relatif au document tels que les conjonctions et les pronoms, l'utilisation de la **racinisation** qui réduit des mots distincts à leur racine grammaticale commune, **L'identification des groupes de noms** ce qui élimine les verbes, adverbes et adjectifs. Des compressions supplémentaires sont applicables. Voir **[6]** chapitre **7**.

1.3.1 Phases de construction de l'index

1.3.1.1.Extraction des termes (Tokenization)

C'est la tâche qui consiste à découper le texte d'un document en pièces (tokens) appelés par abus de langage mots ou termes tout en éliminant éventuellement quelques caractères tels que la ponctuation. Un Token est une instance d'une séquence de caractères dans un document particulier qui sont regroupés comme une unité sémantique utile pour le traitement.

Exemple :

Texte du document :

« L'apprentissage automatique est une branche informatique, celle-ci fait partie de l'intelligence artificielle. »

Résultat d'extraction des tokens :

apprentissage	est	une	branche	informatique	celle	ci	fait	partie	de
intelligence	artificielle								

L'extraction des tokens n'est qu'une étape initiale car souvent les termes de l'index sont assez différents, ceux-là ayant subi une dérivation à partir des tokens.

1.3.1.2 Elimination des mots vides

Parfois, quelques mots extrêmement courants s'avèrent d'une importance symbolique voire nulle dans la sélection des documents qui correspondent aux besoins de l'utilisateur, de ce fait ils sont entièrement exclus du vocabulaire. Ces mots sont appelés **mots vides** (stopwords). La stratégie générale de détermination de la liste des mots vides est de trier les termes par ordre de leurs nombre d'occurrences dans la collection de documents par la suite prendre les termes les plus fréquents filtrés souvent manuellement, ayant peu ou aucun trait avec les thématiques ou domaines des documents à indexer. Les mots de cette liste seront alors éliminés durant l'indexation **[5]**.

La tendance générale des systèmes RI au fil du temps va de l'utilisation standard de listes de mots vides assez volumineuses (200 à 300 termes) à de très petites listes (7 à 12 termes) jusqu'à ne pas y avoir de liste. Les moteurs de recherche Web n'utilisent généralement pas de listes de mots vides. Une partie de la conception des systèmes RI modernes a porté précisément sur la manière dont nous pouvons exploiter les statistiques de la langue afin de pouvoir mieux gérer les mots courants **[5]**.

Exemple de mots vides pour différentes langues :

Langue	Mots vides
Français	De , le , autrefois ,cependant , désormais, compris, car etc.
Anglais	My , our, his, do , while , for, with about, against etc.
Arabe	مثل , حيث , ذلك ,عن ,إلي ,الذي……

Tableau 1 Exemples de mots vides dans différentes langues

1.3.1.3 Normalisation

On présente deux formes de normalisation couramment utilisées et leur mise en œuvre. Dans de nombreux cas, elles semblent utiles, mais elles peuvent dans quelques cas avoir des effets indésirables. En fait, dans la normalisation plusieurs détails peuvent être abordés mais souvent, si le traitement est appliqué de manière cohérente à la requête et aux documents, les détails les plus fins peuvent ne pas avoir beaucoup d'effet global sur les performances.

1. **Prise en charge des Accents et diacritiques :**

Diacritiques sur les caractères en anglais ont une incidence marginale, et nous pourrions bien vouloir faire correspondre cliche et cliché, ou naïfs et naïve. Cela peut être fait en normalisant les tokens pour supprimer les signes diacritiques. Dans de nombreuses autres langues, les diacritiques font régulièrement partie du système d'écriture afin de distinguer les différentes

prononciations. Parfois, les accents distinguent des mots différents. Par exemple, en espagnol, Peña est ‘’une falaise’’, tandis que Pena signifie ‘’chagrin’’.**[5]**

Néanmoins, la question importante est de savoir comment les utilisateurs sont susceptibles d’écrire des requêtes pour ces mots. Dans de nombreux cas, les utilisateurs saisiront des requêtes pour des mots sans diacritiques, pour des raisons de rapidité, de paresse, de logiciels limités ou d’habitude.

2. **Unification de la casse :**

la technique couramment utilisée pour unifier la casse est de tout réduire en minuscule pour faire correspondre ‘’Course ’’ à ‘’course‘’, très utile pour les moteurs de recherche web. D’autre part beaucoup de noms propres sont dérivés de noms communs et ne sont donc distinguées que par le caractère majuscule au début.

En anglais une alternative à la technique sus-décrites est de ne convertir en minuscule que les mots en début de phrases ou dans les titres car en général, ce sont des mots ordinaires.

La normalisation est parfois largement sujette aux spécificités d’une langue comme pour ‘’colour’’ (dictée anglaise) et ‘’color’’ (dictée américaine) ou les pronoms le, la et les en langue française **[5]**.

1.3.1.4 Racinisation et lemmatisation

Cette étape du processus de construction de l’index a pour but de regrouper les variantes d’un mot afin de leur accéder en n’utilisant qu’une seule forme commune.

En effet, pour des raisons grammaticales bien évidentes les mots se présentent dans les documents sous plusieurs formes comme ‘’’acquière’’, ’’acquis’’ et ’’acquisition’’, ou alors avec des dérivations comme pour démocratie, démocratisation etc. Il serait alors intéressant que lors de la recherche d’un de ces mots on retourne les documents contenant un autre mot dans la liste des différentes formes du mot recherché.

La racinisation (stemming) : Troncature de la fin des mots afin de tomber sur une forme commune , incluant souvent l’élimination des affixes de dérivation.

Lemmatisation : Utilisation du vocabulaire et d’analyses morphologiques des mots pour retourner la base du mot ou son entrée de dictionnaire.

Prenons par exemple en anglais, le verbe voir conjugué au passé ‘’saw’’ le stemming peut retourner ‘’s’’ alors que la lemmatisation retournera ‘’see’’ infinitif du verbe voir ou bien ‘’saw’’ tel qu’il est.

L’algorithme de racinisation le plus courant pour l’anglais est l’algorithme de porter, celui-ci consiste en 5 phase de réduction du mot appliquées séquentiellement et pour chacune d’elles il y a des conventions d’application des règles de réduction. Exemple de règles :

Règle	Résultat après réduction	Exemple	Résultat
SSES	SS	Caresses	caress
IES	I	Ponies	poni
SS	SS	Caress	caress
S		Cats	cat

Tableau 2 exemple de règles de réduction pour le stemmer de Porter:

Comparaison entre quelques stemmer pour l'anglais :

Texte original : " *Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation* "

Stemmer de porter: "*such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret*".

Stemmer de lovins : "*such an analys can reve featur that ar not eas vis from th vari in th individu gen and can lead to a pictur of expres that is mor biolog transpar and acces to interpres*"

Stemmer de paice: "*such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret* " **[5]**

1 .3.1.5 Construction de l'index

Après traitement du texte d'une collection de documents on aboutit à une liste finie de termes d'indexation qui constituent le vocabulaire de cette collection. Ensuite on obtient une matrice dite document-termes qui a la forme générale suivante :

	T_1	T_2	T_3
D_1	W_{11}	W_{12}	W_{13}
D_2	W_{21}	W_{22}	W_{32}
D_3	W_{31}	W_{32}	W_{33}

Où :

- D_1 : C'est le premier document.
- T_1 : Le premier terme.
- Wij : poids du terme i dans le document j. ce poids peut être perçu de différentes manières : occurrence ou pas du terme dans le document, nombre d'occurrence, etc.

On peut procéder à l'élaboration de la matrice terme-document qui est l'inverse de la matrice ci-dessus, cette matrice est plus efficace et performante dans le traitement des requêtes car on ne considère que les documents contenant les termes de la requête lors du classement (éventuellement sélection) des documents.

Ces matrices ne constituent qu'une forme générale de l'index. On peut exclure les poids nuls comme on peut ajouter les positions d'un terme dans un document ce qui indique que cette représentation est flexible quant à la détermination des poids et aux propriétés à y ajouter de sorte à adapter l'index au modèle choisi pour effectuer la sélection des documents sensés répondre au besoin de l'utilisateur.

1.4 Modèles de la recherche d'information

La modélisation en RI est un processus complexe ayant pour but de produire une fonction de classement des documents.

Fonction de classement : fonction qui attribue une note à un document vis-à-vis d'une requête, et ce en quantifiant la similarité entre la requête et celui-ci.

Classement : ordre des documents sensé refléter leur pertinence par rapport à une requête utilisateur. Ce problème intègre logiquement un degré d'incertitude.

Selon **[6]** Un modèle de RI est un quadruplé [D, Q, F, R(qi, dj)] où :

- **D :** Ensemble des vues logiques des documents de la collection.
- **Q :** Ensemble des vues logiques des requêtes de l'utilisateur.
- **F :** plate-forme pour la modélisation des requêtes et des documents.
- **R(qi, dj) :** fonction de classement.

1.4.1 Le modèle booléen

C'est un modèle simple qui repose sur la théorie des ensembles et l'algèbre booléenne. Les requêtes y sont exprimées en tant qu'expression booléenne en utilisant des opérateurs booléens tels que : et (^), ou (∨), et la négation (¬).

Pour ce modèle, les poids $\mathbf{W_i}$ dans la matrice terme-document sont tous binaire ce qui veut dire qu'on s'intéresse à l'existence ou non d'un terme dans un document, $\mathbf{W_i} = 1$ signifie l'existence du terme sinon $\mathbf{W_i} = 0$.

Exemple de représentation des requêtes et des documents :

Di = t1 ^ t3 ^ t6.

Qi = t1∨ t6.

Ce modèle cherche la correspondance exacte entre le document et la requête.

Correspondance (d,q) =1 ou 0.

Avantage :

- Assez intuitif avec sémantique précise.
- Formalisme soigné.

Inconvénients :

- Recherche basée sur une décision binaire sans aucun appariement partiel ce qui veut dire que tous les documents retournés sont au même degré d'intérêt donc pas de classement entre les documents.
- Souvent ce modèle retourne peu de documents ou un très grand nombre de documents.
- Les requêtes exprimées dans ce modèle par les utilisateurs sont trop simplistes.

1.4.2 Le modèle vectoriel

D'après **[6]**, à la différence au modèle booléen, le modèle vectoriel n'utilise pas une correspondance booléenne et des poids binaires mais des poids non binaires ce qui lui permet d'offrir une correspondance partielle entre les documents et les requêtes, dès lors il est possible de déterminer un classement entre les documents retournés par le SRI.

Les poids des termes sont utilisés pour calculer un degré de similarité entre les requêtes et les documents, le classement des documents se fait alors par ordre décroissant de cette **similarité**.

- A chaque paire $\mathbf{(t_i,d_j)}$ est associé un poids $\mathbf{W_{ij}}$ **.**
- Les termes de l'index sont supposés être tous mutuellement **indépendants**.
- Les termes sont considérés comme des vecteurs unitaires dans un espace de dimension n qui est le nombre de termes de l'index.
- La représentation d'un document $\mathbf{d_j}$ et d'une requête **q** qui sont des vecteurs n-dimensionnels est comme suit :

- $\overrightarrow{d_j} = (W_{1j}, W_{2j}, \dots, W_{nj})$
- $\vec{q} = (W_{1q}, W_{2q}, \dots, W_{nq})$

La similarité entre dj et q n'est autre que le cosinus de θ qui est l'angle entre les 2 vecteurs :

$\cos\theta = \frac{\overrightarrow{d_j} \bullet \vec{q}}{|\overrightarrow{d_j}| \bullet |\vec{q}|}$.

$$SIM(d_j, q) = \frac{\sum_{i=1}^{n} w_{ij} \, x \, w_{iq}}{\sqrt{\sum_{i=1}^{n} w_{ij}^2} \, x \, \sqrt{\sum_{i=1}^{n} w_{iq}^2}}$$

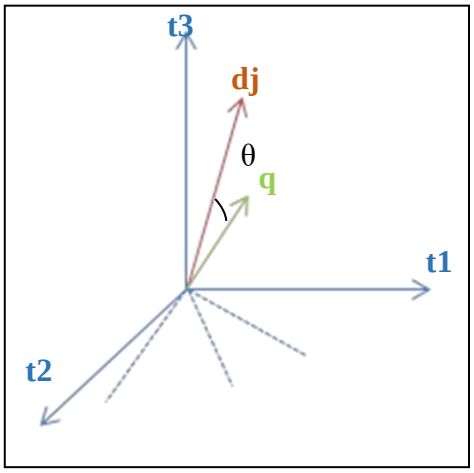


Figure 2 représentation d'un document et d'une requête avec le modèle vectoriel

Comme w_{ij} , $w_{iq} > 0$ alors $0 \leq SIM(d_j, q) \leq 1$.

Les poids W_i sont généralement issus de la pondération TF-IDF.

Pondération TF-IDF :

TF (term frequency): plus un terme est fréquent dans un document plus il est important dans la description de ce document.

DF(document frequency): nombre de document incluant un terme.

IDF (Inverse document frequency) :

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

idf_i : Inverse document frequency du terme i.

N : nombre de documents dans le corpus.

n_i : Fréquence du terme i dans le corpus.

idf fournit une base pour les schémas modernes de pondération des termes et est utilisé pour le classement dans presque tous les SRI.

Les schémas de pondération les plus connus utilisent une combinaison des facteurs idf avec des fréquences de termes.

Soit w_{ij} le poids du terme associé au terme k_i pour le document d_j

Ainsi : $W_{ij} = \begin{cases} (1 + \log f_{ij}) \times \log \frac{N}{n_i} \; si \; f_{ij} > 0 \\ 0 \; sinon \end{cases}$

C'est ce qu'on appelle schéma de pondération TF-IDF.**[6]**

1.4.3 Modèle probabiliste

D'après **[6]**, Le modèle probabiliste schématise le problème de RI en utilisant un cadre probabiliste. Étant donné une requête utilisateur, il existe un ensemble de réponses idéales pour cette requête. Si nous décrivions cet ensemble de réponses idéales, nous pourrions récupérer les documents pertinents. La satisfaction des requêtes passe par une spécification des propriétés de cet ensemble de réponses idéales.
D'abord un ensemble de documents est retourné et l'utilisateur l'inspecte à la recherche des documents pertinents. Le SRI alors utilise cette information pour affiner les attributs des solutions idéales. En répétant ce processus on s'attends à ce que la description des solutions idéales s'améliore.

Le modèle probabiliste :

- Estime la probabilité qu'un document soit pertinent par rapport à une requête utilisateur.
- Suppose que cette probabilité ne dépend que des représentations de la requête et des documents.
- L'ensemble idéale est noté R et celui-ci doit maximiser la probabilité de pertinence

Classement des documents :
Soit :

- R l'ensemble des documents pertinents à la requête q.
- $\bar{R}$ l'ensemble des documents non pertinents à la requête q.
- $P(R|\overrightarrow{dj})$ probabilité que dj soit pertinent à la requête q.
- $P(\bar{R}|\overrightarrow{dj})$ probabilité que dj soit non pertinent à la requête q.

Ainsi la similarité entre un document dj et une requête q est définie comme suit :

$$sim(d_j, q) = \frac{P(R|\overrightarrow{dj})}{P(\bar{R}|\overrightarrow{dj})}$$

Soit :

- N le nombre de documents dans la collection
- n_i le nombre de documents contenant le terme t_i
- R le nombre total de documents pertinents par rapport à la requête q
- r_i le nombre de documents pertinents contenant le terme t_i

On se basant sur ces variables on peut construire la table de contingence suivante :

	Documents pertinents	Documents non pertinents	Tous les documents
Documents contenant t_i	r_i	$n_i - r_i$	n_i
Documents ne contenant pas t_i	$R - r_i$	$N - \text{n i} - (R - r_i)$	$N - n_i$
Tous les documents	R	$N - R$	N

Tableau 3 Table de contingence du modèle probabiliste

Lorsque les informations de la table de contingence sont disponibles pour une requête donnée il est possible d'écrire :

- $p_i R = \frac{r_i}{R}$
- $q_i R = \frac{n_i - r_i}{N - R}$

Ainsi l'équation de calcul du classement des documents pour le modèle probabiliste s'écrit comme suit :

$$sim(d_j, q) \sim \sum_{t_i[q,d_j]} \log \left(\frac{r_i}{R - r_i} \times \frac{N - n\,i - R + r_i}{n_i - r_i}\right)$$

Où $t_i[q, d_j]$ veut dire : $t_i \in q \wedge t_i \in d_j$

Pour gérer les petites valeurs de r_i on ajoute 0.5 à chacun des termes de la formule :

$$sim(d_j, q) \sim \sum_{t_i[q,d_j]} \log \left(\frac{r_i + 0.5}{R - r_i + 0.5} \times \frac{N - n\,i - R + r_i + 0.5}{n_i - r_i + 0.5}\right)$$

Cette formule est considérée comme l'équation de classement classique du modèle probabiliste et est connu sous le nom de : équation de Robertson-Sparck Jones.

Avantages :

- Documents classés par ordre décroissant de leur probabilité de pertinence.

Inconvénients :

- Besoin de définir l'estimation initiale de $p_i R$
- Ne prend pas en compte TF (term frequency) .
- Manque de normalisation de la taille des documents

Comparaison entre les modèles classiques :

- Le modèle booléen est considéré le moins robuste et n'offre pas de fonction de classement pour les documents.

- Il y a controverse sur le meilleur modèle entre vectoriel et probabiliste, W.B.CROFT défend la supériorité du modèle probabiliste alors que Salton a démontré que pour les collections générales le modèle vectoriel est le meilleur, et c'est aussi l'opinion générale au sein de la communauté des chercher en RI.**[6]**

Il existe plusieurs alternatives aux modèles classiques, on peut citer :

- Le modèle vectoriel généralisé
- Modèle des ensembles flous
- Modèle booléen étendu
- Indexation sémantique latente

1.5 Les grands enjeux de recherche d'information

D'après **[4]**, les chercheurs en recherche d'information se sont concentrés sur quelques problèmes clés qui demeurent tout aussi importants à l'ère des moteurs de recherche commerciaux utilisant des milliards de pages Web que lors des tests menés dans les années 1960 sur des collections de documents contenant environ 1,5 Mo de texte. L'une de ces questions est la pertinence.

1.5.1 La pertinence

est un concept fondamental dans la recherche d'informations. En gros, un document pertinent contient les informations recherchées par une personne lorsqu'elle a soumis une requête au moteur de recherche. Bien que cela paraisse simple, de nombreux facteurs entrent en ligne de compte pour déterminer si un document particulier est pertinent ou non. Ces facteurs doivent être pris en compte lors de la conception des algorithmes

Pour comparer du texte et classer des documents. La simple comparaison du texte d'une requête avec le texte d'un document et la recherche d'une correspondance exacte, comme cela peut être fait dans un système de base de données ou à l'aide de l'utilitaire grep sous Unix, produisent des résultats très médiocres en termes de pertinence. Une raison évidente à cela est que le langage peut être utilisé pour exprimer les mêmes concepts de différentes manières, souvent avec des mots très différents.

C'est ce que l'on appelle le problème d'inadéquation du vocabulaire lors de la récupération d'informations. Il est également important de faire la distinction entre pertinence thématique et pertinence utilisateur.

Un document texte est thématiquement pertinent pour une requête s'il porte sur le même sujet. Par exemple, un reportage sur une tornade au Venezuela serait pertinent pour la requête "événements météorologiques violents". La personne qui a posé la question peut ne pas juger le reportage pertinent, toutefois, si elle a déjà vu le récit auparavant, si le récit a cinq ans ou si le récit est en chinois et provient d'une agence de presse chinoise. La pertinence de l'utilisateur prend en compte ces caractéristiques supplémentaires de l'histoire.

Pour résoudre le problème de la pertinence, les chercheurs proposent des modèles de recherche et testent leur fonctionnement. Un bon modèle de RI trouvera les documents susceptibles d'être considérés comme pertinents par la personne ayant soumis la requête. Certains modèles de RI se concentrent sur la pertinence thématique, mais un moteur de recherche déployé dans un environnement réel doit utiliser des algorithmes de classement intégrant la pertinence utilisateur.

Une caractéristique intéressante des modèles de récupération utilisés dans la récupération d'informations est qu'ils modélisent généralement les propriétés statistiques du texte plutôt que la structure linguistique. Cela signifie, par exemple, que les algorithmes de classement sont généralement beaucoup plus concernés par le nombre d'occurrences de mots que par le fait qu'il s'agisse d'un nom ou d'un adjectif. Des modèles plus avancés intègrent des fonctionnalités linguistiques, mais ils ont tendance à être d'importance secondaire. L'utilisation d'informations sur la fréquence des mots pour représenter un texte a commencé avec un pionnier de la RI, H.P. Luhn, dans les années 1950. Cette vision du texte n'est devenue populaire dans les autres domaines de l'informatique, tels que le traitement du langage naturel, jusqu'aux années 1990.

1.5.2 L'évaluation

C'est un autre problème fondamental pour la recherche d'informations. Étant donné que la qualité du classement d'un document dépend de l'adéquation avec les attentes d'une personne, il était nécessaire dès le début de développer des mesures d'évaluation et des procédures expérimentales d'acquisition de ces données et de les utiliser pour comparer les algorithmes de classement. Cyril Cleverdon a ouvert la voie au développement de méthodes d'évaluation au début des années 1960 et Deux des mesures qu'il a utilisées, précision et rappel, sont toujours populaires. La précision est une mesure très intuitive et représente la proportion de documents pertinents par rapport à ceux retournés. Le rappel est la proportion de documents pertinents qui sont récupérés par rapport au total des documents pertinents dans la collection. Lorsque la mesure de rappel est utilisée, il est supposé que tous les documents pertinents pour une requête donnée sont connus. Une telle hypothèse est clairement problématique dans un environnement de recherche Web, mais avec des collections de documents de test plus réduites, cette mesure peut être utile. Une collection de tests pour des expériences de RI consiste en une collection de documents texte, un échantillon de requêtes classiques et une liste de documents pertinents pour chaque requête (les jugements de pertinence}. Les collections de tests les plus connues sont celles associées au forum d'évaluation TREC6.

L'évaluation des modèles de RI et des moteurs de recherche est un domaine très actif, l'accent étant actuellement mis sur l'utilisation de grands volumes de données de journaux provenant d'interactions utilisateur, telles que les données de clic, qui enregistrent les documents sur lesquels vous avez cliqué au cours d'une session de recherche. Les clics et autres données de journaux sont fortement corrélés avec la pertinence, elles peuvent donc être utilisées pour évaluer la recherche.

1.5.3 Le besoin des utilisateurs

La troisième question essentielle en matière de recherche d'informations **est l'accent mis sur les utilisateurs et leurs besoins en information**s. Cela devrait être clair étant donné que l'évaluation de la recherche est centrée sur l'utilisateur. Autrement dit, les utilisateurs d'un SRI sont les juges ultimes de la qualité. Cela a conduit à de nombreuses études sur la façon dont les gens interagissent avec les moteurs de recherche et, en particulier, au développement de techniques pour aider les gens à exprimer leurs besoins en information. Un besoin d'information est la cause sous-jacente de la requête qu'une personne soumet à un moteur de recherche. Contrairement à une demande à un système de base de données, comme pour le solde d'un compte bancaire, les requêtes de texte sont souvent de mauvaises descriptions de ce que l'utilisateur veut réellement. Malgré leur manque de précision, les requêtes en un mot sont très courantes. Dans la recherche Web. Des techniques telles que l'expansion de requête, et la rétroaction de pertinence utilisent l'interaction et le contexte pour affiner la requête initiale afin de produire des listes mieux classées.

2. Intelligence des essaims

2.1 Introduction à l'intelligence des essaims

Selon **[3]**,**[7]** et **[25]:** Historiquement, la nature a été source d'inspiration pour la communauté scientifique dans tous les domaines et a contribué au déclic qui a permis de débloquer des situations et résoudre des problèmes, jadis difficiles.

Le domaine informatique ne constitue pas une exception à ce fait, en se basant sur des observations de systèmes naturels à la fois complexes, optimisés et réussies, de nouvelles approches ont émergé donnant naissance à de nouveaux algorithmes et méthodes qui apportent différents niveaux d'optimisation. L'ensemble des travaux informatiques s'inspirant de la nature a induit la genèse de l'informatique bio-inspirée.

Les essaims dans la nature, tels que les colonies de Fourmies, Les troupes de loups, les colonies d'abeilles et les essaims d'oiseaux partagent tous des caractéristiques intéressantes telles que:

- **L'auto-organisation :** les chemins aux solutions (sources de nourriture par exemple) ne sont pas prédéfinis mais le fruit d'organisation mutuelle et d'adaptation.
- **La décentralisation :** autonomie des individus de la colonie tout en poursuivant les mêmes finalités.
- **La flexibilité :** l'essaim peut gérer les perturbation internes ainsi que les dangers et défis externes (assurer la nourriture...).
- **La robustesse :** les taches sont assurées même si certains individus y échouent (mort d'un individu, …)

Ces caractéristiques permettent à des individus, bien qu'individuellement vulnérables, de collaborer autours de buts communs et exécuter des tâches complexes nécessaires à leur survie et prospérité.

Ce sont ces caractéristiques qui constituent la base de l'inspiration, car certaines tâches informatiques complexes nécessitent un certain degré d'exigence en termes de coordination de partage de ressources et de flexibilité et c'est pour cela que des algorithmes sont conçus pour mimer les comportements des individus d'un essaim et ouvrir de nouveaux horizons dans l'optimisation. Les algorithmes les plus populaires sont l'algorithme d'optimisation par colonies de Fourmies (Ant Colony Optimization – ACO) l'optimisation par les essaims particulaires (Particle Swarm Optimization – PSO) et enfin l'algorithme des abeilles. Ces algorithmes sont hautement performants grâce à leur évolutivité leur apprentissage collectif et aux possibilités de parallélisation des traitements pour les problèmes nécessitant une réponse en temps réel.

Ainsi l'intelligence des essaims (Swarm intelligence) peut être définie comme suit :

Définition 1

‘’*La discipline d'intelligence artificielle qui étudie les systèmes naturels et artificiels de plusieurs individus (agents) agissant en groupes (essaims), exécutant des tâches simples, de façon autonome mais coordonnées produisant ainsi une intelligence collective nécessaire à la résolution de problèmes complexes et à l'optimisation* ‘’, **[25].**

Définition 2

‘’ *toute tentative de conception d'algorithmes ou de dispositifs de résolution de problèmes répartis inspirés du comportement collectif des colonies d'insectes sociaux et d'autres sociétés animales*’’ **[3]**

2.2 Optimisation par colonie d'abeilles artificielles

L'optimisation par colonies d'abeilles artificielles est à la base une optimisation numérique **inspirée** par le **comportement** des abeilles. Elle a démontré son efficacité dans la détermination d'un optimum globale pour les fonctions numériques. Des adaptations de ces techniques ont été apportées pour étendre le champ d'application de cet algorithme.

2.2.1 Source d'inspiration naturelle

D'après **[8]** Les composantes du modèle naturel ayant conduit à l'émergence de l'intelligence collective des abeilles sont : les sources de nourriture, les abeilles employées et les non employées, le modèle a aussi définit 2 types de comportements : le recrutement vers une source de nectar et l'abandon d'une source de nectar complètement exploitée.

1. **La source de nourriture :** La valeur d'une source de nourriture est définie selon plusieurs facteurs à savoir : la proximité par rapport au nid d'abeilles, la concentration en nectar et la facilité d'extraire ce nectar.
2. **Les abeilles employées :** Chacune d'elle se charge de l'exploitation d'une source de nourriture déjà découverte et lorsqu'elle ramène le nectar vers le nid, fait savoir les abeilles présentes sur la profitabilité de cette source, elle indique aussi le chemin à cette source.
3. **Les abeilles non employées :** celles-ci sont toujours à l'affut de sources de nectar à exploiter. Ces abeilles sont de 2 types :
    1. **Scoutes :** elles explorent l'environnement autour du nid en quête de nouvelles sources de nectar.
    2. **Spectatrices :** elles attendent dans le nid afin de recevoir l'information de la part des employées sur les sources de nourriture qu'elles exploitent.

L'échange d'information entre les abeilles est le facteur le plus important dans la construction du savoir collectif

Ce partage se fait dans la zone de danse et chacune des abeilles employées effectue une danse proportionnelle à la valeur de la source de nourriture (richesses, distance et facilité d'exploitation) actuellement exploitées par elles, ainsi la source de nourriture la plus riche a plus de probabilité de voir des abeilles recrutées vers son voisinage.

L'auto-organisation des abeilles repose sur les concepts suivants :

1. **Le feedback positif :** l'information partagée sur les sources de nourriture génère un recrutement vers les meilleures sources.
2. **Le feedback négatif :** Un site épuisé voit son exploitation stoppée et abandonnée
3. **Les fluctuations :** les abeilles scoutes utilisent un processus de recherche aléatoire afin d'explorer de nouvelles sources de nourriture.
4. **Les multiples interactions:** Utilisation de la zone de danse pour l'échange d'informations.

2.2.2 Algorithme de colonie d'abeilles artificielles

Selon **[8]**, Grâce aux concepts naturels décrits ci-dessus, un nouvel algorithme : ''Artificial bee colony algorithm'' (ABC) simulant le comportement d'abeilles réelles et particulièrement les mécanismes de recherche de nourriture, est décrit pour la résolution de problèmes multidimensionnels. Le model la colonie comporte trois groupes d'abeilles : employées, spectatrices et scoutes la première moitié de la colonie est constituée des abeilles employées artificielles et la deuxième moitié se sont les spectatrices. Pour chaque source de nourriture il n'y a qu'une seule abeille employée c'est-à-dire que le nombre de sources de nourriture est égal au nombre d'abeilles employées. Les employées dont la source de nourriture est épuisée par les abeilles devient scoute.

Pseudo-code général de l'algorithme des abeilles artificielles ABC:

Début

Envoyer les abeilles scoutes vers les sources de nourritures initiales.

Répeter

Envoyer les abeilles employées vers les sources de nourriture et déterminer leur teneur en nectar ;

Calculer la probabilité avec laquelle les abeilles spectatrices choisissent leurs sources de nourriture ;

Arrêter l'exploitation des sources de nourritures abandonnées par les abeilles ;

Envoyer les abeilles scoutes vers la zone de recherche afin de découvrir de nouvelles sources de nourriture aléatoirement ;

Jusqu'à conditions vérifiée

Fin

Chaque cycle de recherche comprend trois étapes: déplacer les abeilles employées et spectatrices vers les sources de nourriture et calculer leur quantité de nectar; et déterminer les abeilles scoutes et les diriger vers les sources possibles de nourriture. Une position de **source de nourriture** représente une **solution possible** au problème à optimiser. La **quantité de nectar** d'une source de nourriture correspond à la **qualité de la solution** représentée par cette source de nourriture. Les spectatrices sont placées sur les sources de nourriture en utilisant un processus de sélection basé sur les probabilités. À mesure que la quantité de nectar d'une source de nourriture augmente, la probabilité avec laquelle la source de nourriture est préférée par les spectatrices augmente également. Chaque colonie d'abeilles a des scoutes qui sont ses explorateurs. Aux explorateurs (abeilles scoutes) on n'exige pas trop il suffit qu'ils trouvent des sources de nourriture peu importe le type ou la quantité. En raison de ce comportement, les scoutes se caractérisent par de faibles coûts de recherche et une faible qualité moyenne de la source de nourriture. Dans le cas des abeilles artificielles, les scoutes artificielles pourraient avoir pour tâche de découvrir rapidement le groupe de solutions réalisables. Dans ce travail, une des abeilles employées est sélectionnée et classée comme abeille scoute. La sélection est contrôlée par un paramètre de contrôle appelé "**limite**". Si une solution représentant une source de nourriture n'est pas améliorée par un nombre d'essais prédéterminé, cette source de nourriture est abandonnée par l'abeille employée et celle-ci est reconvertie en scoute. Le nombre d'essais avant l'abandon d'une source de nourriture est égal à la valeur de "limite", qui est un paramètre de contrôle important de l'ABC.

Dans un processus de recherche robuste, les processus d'exploration et d'exploitation doivent être exécutés ensemble. Dans l'algorithme ABC, tandis que les spectatrices et les abeilles employées effectuent le processus d'exploitation dans l'espace de recherche, les scoutes contrôlent le processus d'exploration.

Dans le cas de véritables abeilles à miel, le taux de recrutement représente une «mesure» de la rapidité avec laquelle l'essaim d'abeilles localise et exploite la source de nourriture

récemment découverte. Le processus de recrutement artificiel pourrait également représenter la «mesure» de la rapidité avec laquelle les solutions réalisables ou les solutions optimales des problèmes d'optimisation difficiles peuvent être découvertes. La survie et le progrès de l'essaim d'abeilles réel dépend de la découverte rapide et de l'utilisation efficace des meilleures ressources alimentaires. De même, la solution optimale des problèmes d'ingénierie difficiles est liée à la découverte relativement rapide de « bonnes solutions », en particulier pour les problèmes nécessitant une résolution en temps réel.

2.2.3 Correspondance entre les concepts réel et artificiels pour les colonies d'abeilles

Concept réel	Correspondance artificielle
Abeille	Agent ,programme, objet
Source de nourriture	Solution potentielle du problème
Quantité de nectar	Qualité de la solution trouvée
Fitness	Mesure de la qualité de la solution
Taux de recrutement des abeilles	Probabilité de génération de solutions dans le voisinage d'une solution en mémoire et aussi mesure de convergence de l'algorithme vers de bonnes solutions.
Exploitation	Déterminer la qualité d'une solution.
Exploration	Trouver des solutions potentielles dans l'espace de recherche.
Epuisement d'une source de nourriture	Pas d'amélioration de la qualité de la solution au bout de d'un nombre d'itérations = **Limite**

Tableau 4: Correspondance entre concepts réels et artificiels pour les colonies d'abeilles

3. Apprentissage automatique

3.1 Introduction à l'apprentissage automatique

L'apprentissage automatique (AA) est de nos jours une discipline incontournable afin d'offrir des solutions novatrices et révolutionnaires dans plusieurs domaines et apporter des éléments qui permettent l'anticipation du changement dans un environnement assez dynamique et intermittent.

Il s'apparente à un bébé qui à partir de ses observations quotidiennes et sans que quelqu'un ne lui consacre du temps pour lui apprendre arrive à analyser son environnement et acquérir des

connaissances telles que la capacité de parler et d'utiliser des formules linguistiques complexes ainsi que des capacités sociales et émotionnelles. Il peut aussi facilement reconnaitre les différentes espèces d'animaux. Il a un apprentissage autonome qui lui confère un savoir et des capacités cognitives qui lui permettent d'interagir avec son environnement.

Même chose pour l'apprentissage automatique, la machine est dotée d'une capacité qui lui permet d'exploiter des milliers voir des millions d'observations sous forme de données en entrée (textes, vidéos, datasets, images …) afin de déceler des tendances et des règles qui permettent à la machine de mieux comprendre les différentes situations, résoudre des problèmes et prédire l'avenir voir même simuler l'intelligence humaine.

Définitions de l'apprentissage automatique :

Même au sein de la communauté scientifique spécialisée en apprentissage automatique, il n'y a pas une définition qui fasse l'unanimité mais chacun sa vision de ce qu'est l'AA voici quelques-unes:

Définition 1 :

'' *L'apprentissage automatique c'est la science qui permet à l'ordinateur d'apprendre et agir comme l'être humain et d'améliorer son apprentissage au cours du temps de façon autonome, en lui injectant des données et des informations sous forme d'observations et d'interaction du monde réel.* '' **[11]**

Définition 2 :

'' *On dit qu'un programme d'ordinateur apprend de l'expérience E avec respect à une classe de tâches T et à la mesure de performance P, si ses performances aux tâches T, mesurées par P, s'améliore avec l'expérience E.* **''** **[2]**

Définition 3 :

'' *L'apprentissage automatique est la science qui donne aux ordinateurs la faculté de réagir dans différentes situations sans en être explicitement programmé pour le faire.*'' **Arthur Samuel**

3.2 Applications de l'apprentissage automatique

L'apprentissage automatique est en permanente évolution et chaque année émanent de nouvelles inventions approches et utilisations qui le rend incontournable dans la vie moderne des individus et des différentes organisations.

Il existe plusieurs domaines et secteurs (Quasiment tous les domaines peuvent en faire appel) où l'apprentissage automatique est utilisé et ce de manière plus ou moins significative parmi ce là :

- Santé

- Sécurité
- Information
- Economie

Exemples d'application de l'apprentissage automatique **[W2]**:

- Traduction automatique
- Filtrage collaboratif
- Reconnaissance d'identité
- Reconnaissance de voix
- Reconnaissance faciale

Ceci ne constitue qu'un très bref aperçu sur ce que l'apprentissage automatique peut apporter aux différents acteurs dans différents domaines économie, industrie, etc…

3.3 Approches d'apprentissage automatique

Il existe différents types d'apprentissage automatique, cela dépend du type de problème à résoudre et du type et volume de données à exploiter.

3.3.1 Apprentissage supervisé

Il repose sur un jeu de données à disposition et une certaine compréhension de ces données.

Le but est qu'à partir de ces données essayer de détecter un modèle qui serait suivi par la majorité des données afin de pouvoir utiliser ce dernier pour prédire au cas où de nouvelles données arrivent, la classe d'appartenance de ces données.

Chaque instance (exemple) de ces données est appelé observation et est décrite par un ensemble d'attributs dont la classe à laquelle elle appartient, cette classe est souvent appelé étiquette.

Le jeu de données est réparti en deux :

- Un ensemble d'apprentissage (aux alentours de 80% des données) : celui-ci est utilisé pour construire un modèle.
- Un ensemble de test : celui-ci est utilisé pour valider le modèle construit.

Dans l'étape de test si on obtient un taux d'erreur acceptable par rapport au résultat déjà connu (étiquettes de chacun des exemples) on peut dire que le modèle est valide et s'en servir pour la prédiction.

Si le modèle construit obtient un taux d'erreurs faible pour les exemples d'entrainement mais n'arrive pas à prédire l'étiquette lors de la phase de test on parle de sur-apprentissage (overfitting) cela voudrait dire que ce modèle est très personnalisé par rapport aux données d'entrainement et échouera dans la prédiction pour un jeu de données plus large.

Lorsque les étiquettes sont des valeurs numériques on parle de régression sinon si l'ensemble de valeurs que ça peut prendre est fini on parle de classification.

Exemple : reconnaissance faciale dans une entreprise.

Données	Ensemble d'images faciale du personnel de la société dont plusieurs images pour la même personne dans différentes situations.
Attribut pour chaque exemple	Caractéristique de l'image : pixels, luminosité ….
Etiquettes pour chaque exemple (personne)	Nom de la personne

Tableau 5 Exemple d'apprentissage supervisé

3.3.2 Apprentissage non supervisé

L'apprentissage non supervisé est utilisé lorsque le problème à résoudre nécessite la manipulation d'un volume de données important, non étiqueté et non classifié tel que les données des réseaux sociaux **[1]**.

La compréhension du sens derrière ces données nécessite des algorithmes capables d'identifier les motifs et les classes et classer les données en fonction de ceux-là et ce à partir des similitudes et différences.

L'apprentissage non supervisé est entre autres utilisé dans la détection de spam dans les courriels, En effet il y a beaucoup de variables dans les e-mails réguliers et les spams pour un analyste afin de marquer le courriel indésirable, d'où l'intérêt des algorithmes de segmentation et d'association **[1]**.

L'apprentissage automatique segmente les données en groupes d'exemples ou groupe de variables.

3.3.3 Apprentissage par renforcement

C'est un modèle d'apprentissage comportemental basé sur une récompense (positive ou négative) attribuée à chaque fois qu'une action fructueuse par rapport à un but fixé est entreprise par un programme (agent) ou robot et ce dans un environnement dynamique. Par conséquent, une séquence de décisions réussies Cela aura pour effet de maximiser la récompense perçue et donc « **renforcer** » le processus car il résout au mieux le problème posé **[1]**.

3.3.4 Apprentissage profond

L'apprentissage profond est une technique d'apprentissage automatique qui inclue les réseaux de neurones sous forme de couches successives appelées couches cachées afin d'apprendre à partir des données de manière itérative. Il est particulièrement utile pour apprendre à partir de données non structurées.

L'apprentissage profond comprend une couche d'entrée, des couches intermédiaires cachées et une couche de sortie **[1]**.

Les données sont injectées à partir de la couche d'entrée. Elles sont ensuite modifiées dans les couches cachées ainsi que dans la couche de sortie en ajustant les poids appliqués à ces nœuds.

En adoptant une approche itérative l'apprentissage profond ajuste et fait des inférences jusqu'à atteindre un point d'arrêt.

L'apprentissage profond est une technique qui utilise une combinaison d'algorithmes supervisés et non supervisés. Plus le problème est complexe plus il y a de couches cachées.

3.4 Classification

Etant donné une nouvelle observation (donnée) dont on ne connait pas le groupe auquel elle appartient, la classification intervient à l'aide de différentes techniques pour prédire la classe d'appartenance de cette nouvelle donnée.

Classe : identification de groupe avec des caractéristiques qui leur sont propres.

La classification est un problème d'apprentissage supervisé car les classe sont connues à l'avance.

Qualité de classification : elle repose sur la minimisation de l'erreur de prédiction

Phases du processus de classification :

1. **Phase de construction du modèle.**
2. **Phase de test :** Tester la précision du modèle par rapport aux données de test et s'en servir pour la classification de nouvelles données.

Critères d'évaluation des méthodes de classification :

- Précision (taux d'erreur acceptable).
- Temps d'exécution.
- Robustesse (insensibilité aux données manquantes et au bruit).
- Possibilités de généralisation.
- Simplicité.

Différentes techniques de classification :

- Classification bayésienne
- Arbres de décision
- Réseaux de neurones
- KNN (K plus proches voisins)

3.5 Clustering (la segmentation)

Selon **[11]**, le clustering est l'une des techniques d'apprentissage non supervisé et est parmi les plus largement utilisées pour **l'analyse exploratoire de données**. Dans toutes les disciplines, des sciences sociales à la biologie, en passant par l'informatique, les gens essaient de se faire une première idée de leurs données en identifiant des groupes significatifs parmi les enregistrements de données. Par exemple, les biologistes informatiques regroupent des gènes sur la base de similitudes dans leur expression dans différentes expériences; les détaillants regroupent les clients, sur la base de leurs profils, à des fins de marketing ciblé et les astronomes regroupent des étoiles sur la base de leur Proximité spatiale.

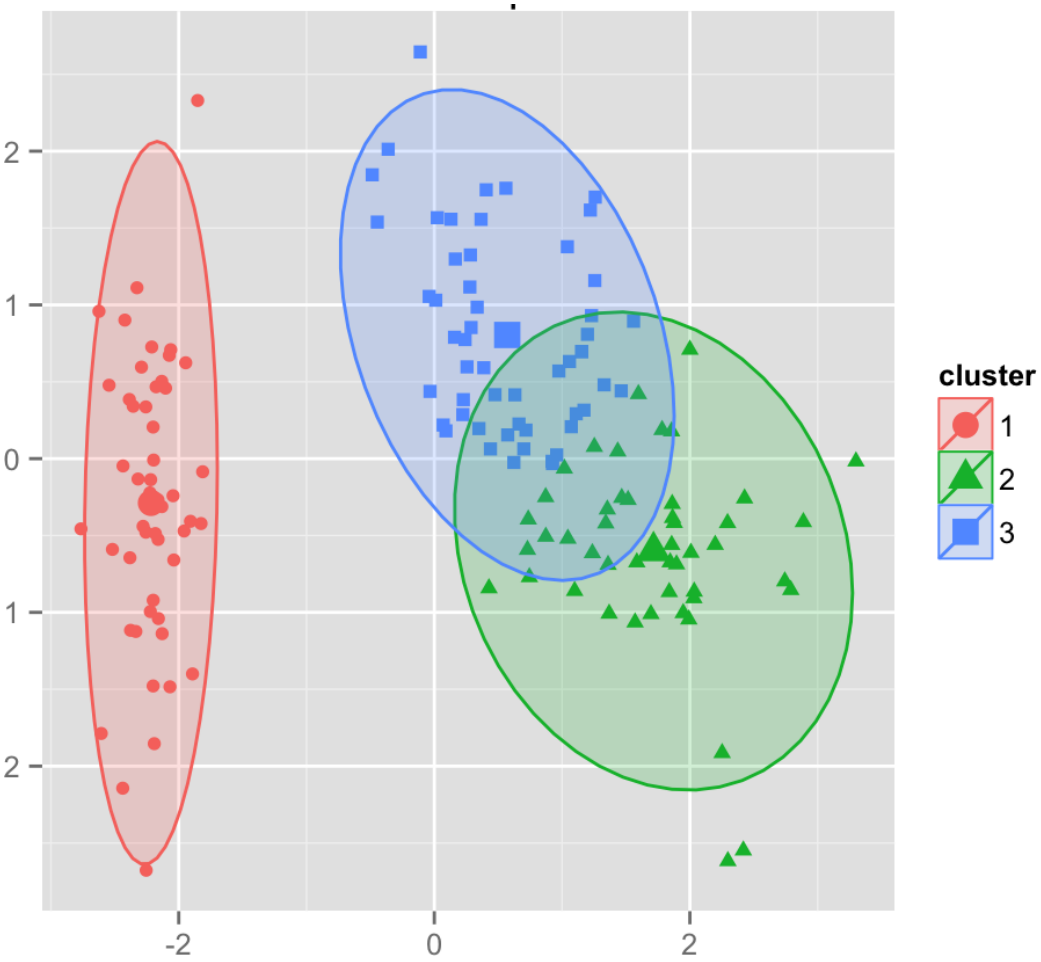


Figure 3 : Illustration de clustering

Le premier point à clarifier est, naturellement, qu'est-ce que le clustering?

Intuitivement, la mise en cluster consiste à :

1. **Regrouper** un ensemble d'objets de telle sorte que des **objets similaires** se retrouvent dans le même groupe.
2. **Séparer** les objets tel que ceux **dissemblables** soient en différents groupes.

Tels sont les critères essentiels pour déterminer la qualité d'un clustering.

A première vue, cette description est assez imprécise et peut-être ambiguë car il n'est pas du tout évident de parvenir à une définition plus rigoureuse.

Il existe plusieurs sources pour cette difficulté. Un problème fondamental est que les deux objectifs mentionnés dans la déclaration précédente peuvent souvent se contredire **[11]**. Mathématiquement, la similarité (ou la proximité) n'est **pas** une relation **transitive**, alors que l'appartenance à un cluster est une relation d'équivalence et, en particulier, une relation transitive. Plus concrètement, il se peut qu'il existe une longue séquence d'objets, $X_1, \dots, .., X_m$ tels que chaque X_i est très similaire à ses deux voisins, $X_i - 1$ et $X_i + 1$, mais X_1 et X_m sont très dissemblables. En plaçant cette séquence dans le même cluster on porte atteinte à la deuxième condition.

La réalisation de notre objectif se résume à maximiser l'inertie inter-classe (entre deux instances de données de groupes différents) et minimiser l'inertie intra-classe.

1. Inertie totale = $\sum_{i=1}^{n} \frac{d(X_i,g)^2}{n}$
   n : nombre d'instances de la population.
   x : valeur de l'instance de données.
   g : Centre de gravité de la population.
   d : fonction de distance.
2. Inertie intra-classe = $\sum_{i=1}^{c} Inertie_i$
   $Inertie_i$: inertie totale de la classe i .
   c : nombre de classes.

Inertie inter-classe : $\sum_{i=1}^{c} p_i * \ d(g_i, g)^2$:

p_i : Probabilité de la classe (proportion du nombre d'individus de la classe par rapport au nombre total d'individus).

D'après **[11],** Un autre enjeu pour le clustering c'est qu'au final, il n'y a pas de certitude quant à l'évaluation du succès du clustering. Même sur la base d'une connaissance complète de la distribution de données sous-jacente, il n'est pas clair quelle est la segmentation '' correcte '' pour ces données ou comment évaluer une segmentation proposée. En effet, il est possible de classer un ensemble d'objets de manières différentes tout en étant toutes significatives. En raison de cela, il y une variété d'algorithmes produisant chacun un résultat différent selon le besoin.

Technique des K-means :

Elle consiste à répartir l'ensemble des enregistrements, sur k centres de gravité choisis arbitrairement selon un critère de distance. ainsi un enregistrement $X_i$est affecté au groupe du centre de gravité le plus proche. On recalcule de nouveaux centres de gravité pour chaque groupe et on réaffecte les enregistrements comme précédemment.

Ce procédé est réitéré jusqu'à ce qu'aucun enregistrement ne change d'affectation et que les groupes se stabiliseront.

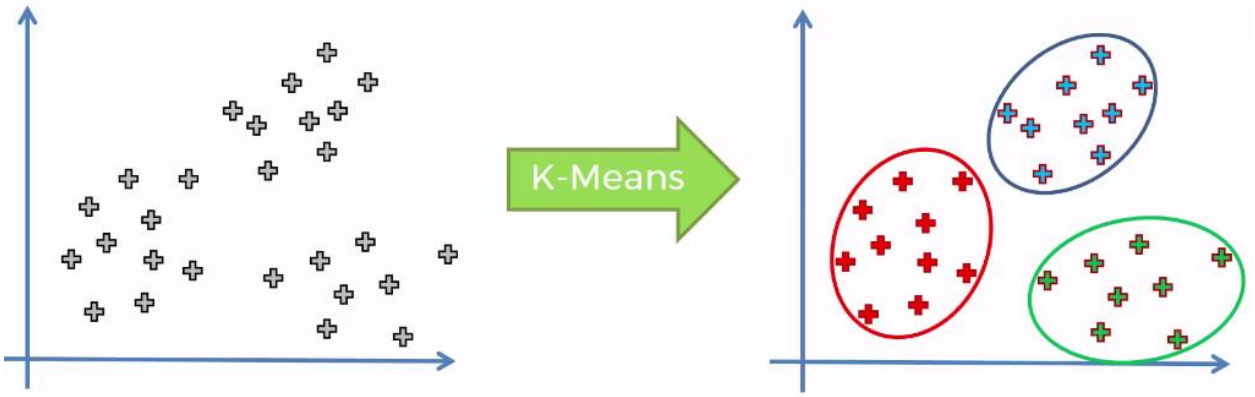


Figure 4: Illustration de l'algorithme K-means

Pseudocode k-means :

Début

Entrées : X $\subset R^n$;nombre de clusters k.

Initialiser aléatoirement les centroïdes initiaux $\mu_1, \mu_2, \ldots \ldots \ldots \ldots, \mu_k$;

Répéter

$\forall i \in [k]$, C_i= {x ∈ X tel que : i = $argmin_j$ ||x − μ j|| }

$\forall i \in [k]$ reclaculer le centroïdes $\mu_i = \frac{\sum_{x \in C_i} x}{|C_i|}$

Jusqu'à convergence.

Conclusion

Des concepts d'intelligence artificielle clés dans la compréhension et l'étude de la problématique ont été présentés à travers ce premier chapitre apportant ainsi plus de clarté au sujet d'étude.

D'autres concepts ont été découvert notamment l'intelligence des essaims et un algorithme sous-jacent qui est l'algorithme des abeilles artificielles dont l'étude est au cœur de ce sujet.

Chapitre 02
Etat de l'art

Introduction

De nombreuses recherches ont été menées au sujet des algorithmes bio-inspirés entre autres les algorithme d'intelligence des essaims, par la communauté scientifique, ainsi beaucoup de variantes et d'adaptations à des situations particulières ont été apportées aux différents algorithmes pour en tirer les meilleurs résultats ou pour des applications bien spécifiques.

Ces techniques sont combinées en cas de besoins avec d'autres techniques d'IA pour mieux rentabiliser leur contribution.

Ces approches trouvent leurs applications dans différents domaines dont la recherche d'information. Ainsi l'étude de cette littérature est une étape essentielle pour la maitrise du domaine de recherche ciblé à travers cette étude.

Nous présentons l'algorithme des abeilles, ses différentes variantes et ses hybridations ensuite nous nous penchant sur les travaux réalisés dans le domaine de la recherche d'information impliquant cet algorithme et les autres techniques d'apprentissage automatique.

1. Utilisation initiale et variantes de l'algorithme des abeilles

La publication en **[8]** a décrit les bases de l'algorithme des abeilles sans trop donner de détails techniques mettant l'accent sur les grandes lignes de l'algorithme, celle-ci constitue une référence et une base d'adaptation dans différents domaines informatiques. Le même auteur a ensuite 2 ans plus tard donné en détail sa technique pour l'optimisation numérique avec et sans contraintes en **[9]**et **[10]** respectivement. Depuis, des variantes de cet algorithme sont apparues pour apporter des solutions novatrices à des problèmes jusqu'ici difficiles à optimiser avec les méthodes classiques.

1.1 L'algorithme ABC Pour les problèmes d'optimisation numériques

C'est le premier cas d'utilisation de l'algorithme des abeilles artificielles**[10]**

Problème d'optimisation global :

Le problème d'optimisation global peut être résumé à la recherche du vecteur $\vec{x}$ qui minimise la fonction objective f($\vec{x}$):

$$minimiser\ f(\vec{x}) = (x_1, x_2, \dots, x_i, \dots, x_{n-1}, x_n) \in R^n \quad (1)$$

Avec (ou sans) les contraintes suivantes :

$$l_i \leq x_i \leq u_i \ \ i = 1, \dots, n \quad (2)$$

Avec la contrainte :

$$g_j(\vec{x}) \leq 0\ pour\ j = 1, \dots, p \quad (3)$$

Et:

$$h_j(\vec{x}) = 0\ pour\ j = p+1, \dots, q \quad (4)$$

$f(\vec{x})$ est définie sur l'espace de recherche S (S $\subseteq R^n$). Les variables sont limitées par les bornes inférieures et supérieures (l_i, u_i) respectivement.

La méta-heuristique des colonies d'abeilles artificielles :

En ABC, la colonie d'abeilles artificielles comprend trois groupes d'abeilles: les abeilles employées associées à des sources de nourriture spécifiques, les abeilles spectatrices observant la danse des abeilles employées dans la ruche pour choisir une source de nourriture et les abeilles à la recherche de sources de nourriture au hasard qui sont les scoutes. Les

spectatrices et les scoutes sont aussi appelés des abeilles non employées. Initialement, toutes les sources de nourriture sont découvertes par les abeilles scoutes. Par la suite, le nectar des sources de nourriture est exploité par les abeilles employées et les abeilles spectatrices, et cette exploitation continuelle finira par les épuiser. Ensuite, l'abeille employée qui exploitait la source de nourriture épuisée devient une abeille éclaireuse à la recherche de sources de nourriture supplémentaires. En d'autres termes, l'abeille employée dont la source de nourriture a été épuisée devient une abeille scoute et va explorer l'espace de recherche en quête d'une nouvelle source de nourriture. Dans l'algorithme ABC, la position d'une source de nourriture représente une **solution possible** au problème et la quantité de nectar d'une source de nourriture correspond à la qualité de la solution associée. Le nombre d'abeilles employées est égal au nombre de sources de nourriture (solutions) puisque chaque abeille employée est associée à une et une seule source de nourriture.

Pseudo-algorithme ABC :

```
Phase d'initialisation
Répéter
  Phase des abeilles employées
  Phase des abeilles spectatrices
  Phase des abeilles scoutes
  Mémoriser la meilleure solution trouvée jusque là
Jusqu'à (nombre d'itération = Max_iter ou temps_maxs)
```

Phase d'initialisation :

Tous les vecteurs de la population de solutions $\vec{x}_m$ sont initialisés (m = 1….SN SN :Nombre de solutions). Chaque vecteur $\vec{x}_m$ possède n variables ($\vec{X}_{mi}$,i=1….n) à optimiser afin de minimiser la fonction objective.

L'équation suivante peut être utilisée pour l'initialisation :

$$\vec{x}_{mi} = l_i + \alpha * (u_i - l_i) \quad (5)$$

Avec α *aléatoire et* $\alpha \in [0,1]$ et u_i, l_i étant les bornes supérieurs et inférieures.

On initialise aussi les **paramètres de contrôle** de l'algorithme tels que :

Max_iter : qui est le nombre maximum d'itérations à effectuer.

Limit : qui est le nombre d'itération sans amélioration de la solution d'une abeille qui lui permet de quitter l'endroit qu'elle exploite.

Phase des abeilles employées :

Les abeilles employées cherche de nouvelles solutions ($\vec{v}_m$) qui seront dans le voisinage des ($\vec{x}_m$) solutions initiales et qui doivent être **meilleures** que celles-ci. Pour le savoir il faut calculer la variable appelée : fitness qui représente la qualité de la solution ou en d'autre termes, la solution ayant minimisé au mieux la fonction objective $f(\vec{x})$ aura la fitness la plus élevée.

Chaque abeille peut générer une solution $\vec{v}_m$ dans le voisinage de $\vec{x}_m$ qu'elle a en mémoire en utilisant l'équation suivante :

$$v_{mi} = x_{mi} + \emptyset_{mi}\,(x_{mi} - x_{ki}) \tag{6}$$

Où $\overrightarrow{x_k}$est choisie au hasard ,i choisi aléatoirement et $\emptyset_{mi}$ un nombre aléatoire dans l'intervalle [-1,1] qui sert à contrôler la production d'une position dans le voisinage de X_{mi}

Après la production d'une potentielle solution, on calcule sa fitness (profitabilité). Ainsi une sélection est effectuée entre $\vec{v}_m$ et $\vec{x}_m$et celle qui a la fitness la plus élevée est gardée.

On peut calculer la fitness d'une solution $\vec{X}_m$comme suit :

$$fit_m(\overrightarrow{x_m}) = \begin{cases} \dfrac{1}{1 + f_m(\overrightarrow{x_m})} & si\ f_m(\overrightarrow{x_m}) \geq 0 \\ 1 + abs(f_m(\overrightarrow{x_m})) & si\ f_m x_m) < 0 \end{cases}$$

Où $f_m(\overrightarrow{X_m})$ est la fonction objective à minimiser.

Phase des abeilles spectatrices :

Les abeilles spectatrices choisissent leur source de nourriture grâce à la probabilité calculée à partir des valeurs de fitness fournies par les abeilles employées.

La probabilité p_m avec laquelle une solution $\overrightarrow{x_m}$ est choisie peut être calculée suivant cette équation :

$$p_m = \frac{fit_m(\overrightarrow{x_m})}{\sum_{z=1}^{SN} fit_z(\overrightarrow{x_z})} \tag{7}$$

Après le choix d'une solution $\overrightarrow{x_m}$, une solution $\vec{v}_m$ dans son voisinage est calculé avec l'équation (6). Et une sélection est effectuée comme susmentionné dans la phase précédente. Ainsi plus d'abeilles sont recrutées vers les sources les plus riches (vers le voisinage susceptible de produire la meilleure solution, celle qui minimise la fonction objective).

Phase des abeilles scoutes :

Les abeilles employées dont les solutions ne peuvent pas être améliorées après un nombre prédéterminé d'essais, spécifié par l'utilisateur de l'algorithme ABC via la variable de contrôle ‘’limit’’ qui constitue le critère d'abandon d'une solution, deviennent des scoutes et leurs solutions sont abandonnées. Ensuite, ces scoutes reconverties commencent à rechercher de nouvelles solutions, de manière aléatoire. Comme dans la phase d'initialisation et ce en utilisant l'équation (5). Par conséquent, les sources initialement pauvres ou qui ont été rendues pauvres par l'exploitation sont abandonnées et un comportement de rétroaction négative apparaît pour équilibrer la rétroaction positive.

1.2 Illustration de la convergence de l'algorithme des abeilles pour l'optimisation numérique

Voici une illustration des différentes itérations qui mènent l'algorithme à la convergence :

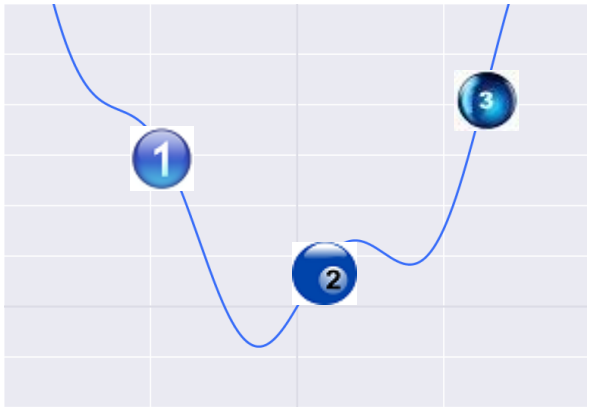


Figure 1: Phase d'initialisation

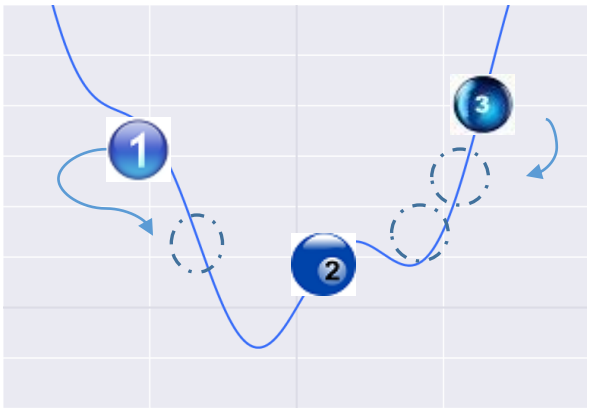


Figure 2: 1è Itération: phase des abeilles employées

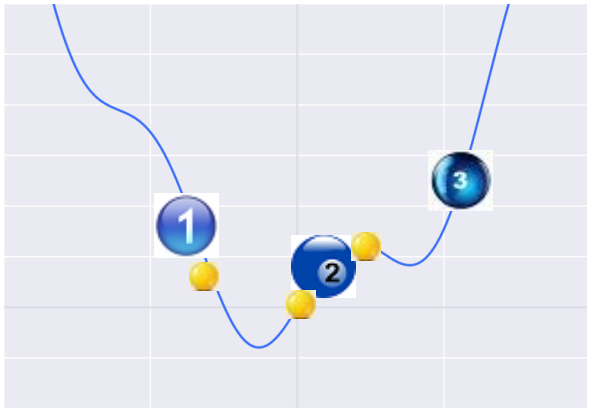


Figure 3: Itération1: Phase des abeilles spectatrices

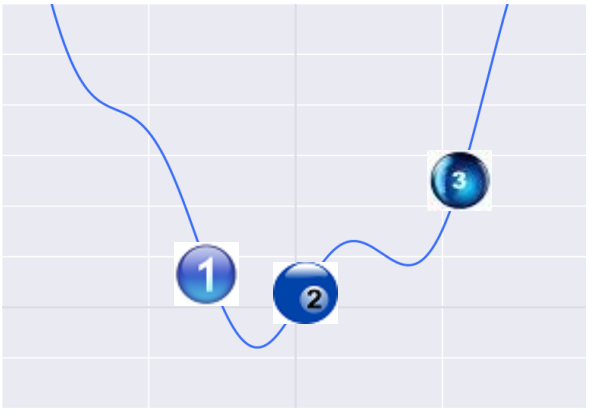


Figure 4:Itération1: phase des abeilles scoutes

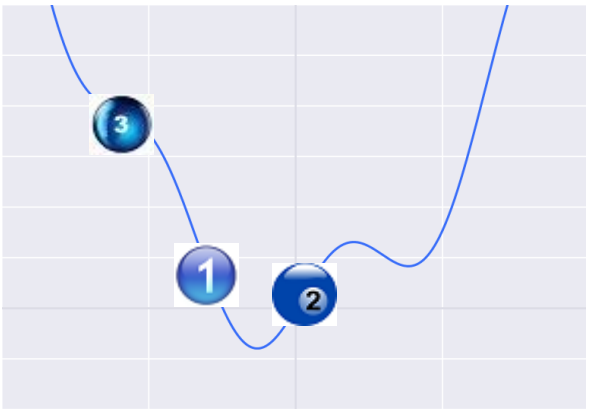


Figure 5 abandon de la source de l'abeille scoute

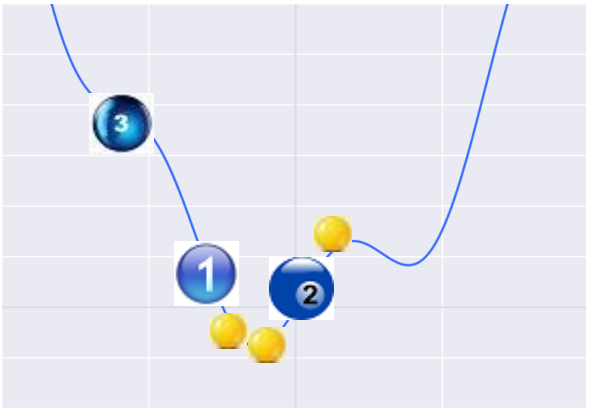


Figure 5 recrutement d'abeilles spectatrices

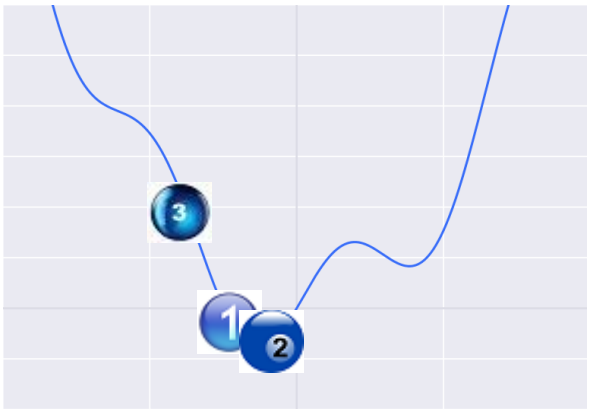
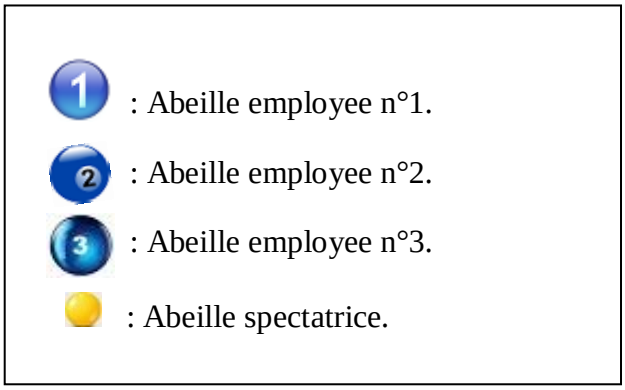


Figure 7: Convergence vers le minimum global



Explication des différentes étapes:

Figure	Explication
Figure1	On place trois abeilles employées (solutions) arbitrairement dans l'espace de recherche.
Figure2	Chaque abeille génère une solution dans son voisinage.et les abeilles 1 et 3 se déplacent vers ces nouvelles solutions car elle sont meilleures, par contre l'abeille n°2 garde sa solution car elle n'est pas améliorée.
Figure3	3 abeilles spectatrices sont recrutées 2 d'entre elles vers le voisinage de l'abeille 2 car c'est celle qui a la meilleure fitness étant celle la plus proche de l'optimum et l'autre vers le voisinage de l'abeille 1.
Figure4	Les abeilles 1 et 2 migrent vers leurs spectatrices ayant découvert de meilleures solutions que les siennes. L'abeilles 3 reste automatiquement surplace.
Figure5	A la prochaine itération l'abeille 3 quitte l'endroit qu'elle exploite car pas de meilleure solution et génère une nouvelle position dans l'espace de recherche

Figure6	Les spectatrices sont redistribuées et elle trouvent de nouvelles solutions
Figure7	Les abeilles employées migrent vers les solutions nouvellement découvertes

Tableau 6: Explication des étapes de convergence de l'algorithme des abeilles

1.3 Variantes de l'algorithme des abeilles

De plus en plus de chercheurs se penchent sur l'utilisation de l'algorithme des abeilles pour la résolution de différents problèmes grâce aux améliorations qu'il peut offrir, ainsi chacun d'eux y apporte sa touche personnelle en l'adaptant au mieux à la problématique qu'il essaye d'étudier et en résultat, une multitude de variantes de cet algorithme ont été conçues et testées.

Recherche chaotique ABC (CABC)

La recherche locale de ABC autour de la meilleure solution est toujours limitée et donc dans [12] la méthode de recherche chaotique est utilisée pour résoudre ce problème. Dans le CABC, les abeilles spectatrices appliquent une séquence chaotique pour améliorer le comportement de recherche local et éviter d'être coincées dans un optimum local. Pour les abeilles spectatrices, la séquence chaotique est enregistrée dans la source de nourriture. Les abeilles spectatrices prennent une décision entre l'ancienne source de nourriture et la nouvelle source de nourriture selon une stratégie de sélection gourmande. Le système de chaos utilisé est défini par l'équation :

$$X_{i+1} = \mu * x_i * (1 - x_i) \quad (1)$$

où μ est un attracteur chaotique. Si μ est égal à 4 qui est un paramètre convenu, le système entre dans un état de chaos total et x_{i+1} est la valeur de la variable x_i en itération. La nouvelle source de nourriture sera calculée comme dans l'équation :

$$x = x_{mi} + R * (2 * x_i - 1)$$

où x est la nouvelle source de nourriture; x_{mi} est le centre de la recherche locale; la position d'abeille sélectionnée, x_i, est la variable de chaos calculée comme dans l'équation (1) ci-dessus; et R est le rayon de la nouvelle source de nourriture générée :

ABC Parallèle :

Une version de l'ABC qui offre un passage de messages parallèles est proposé en [13] pour améliorer sa performance.

Leur approche proposée utilise le programme à chargement statique où ils ont calculé la quantité de tâches assignées requises sur chaque nœud, car la quantité de tâches assignées sur

chaque nœud est proportionnelle à la capacité de calcul du nœud. Ainsi, chaque nœud s'attend à terminer la tâche en même temps, minimisant ainsi le temps d'exécution du programme.

La version parallélisée a été formulée comme suit:

Étape 1 : Initialiser l'environnement d'exécution.

Étape 2 Si le processus actuel est le processus principal, obtenir des algorithmes liés aux paramètres initiaux tels que la taille de la population, envoyer les paramètres susmentionnés et diviser l'échantillon sur chaque processus esclave. Chaque processus génère de manière aléatoire l'emplacement initial du nectar constitué de n solutions.

Étape 3 Utiliser les abeilles de chaque processus conformément aux formules de la recherche standard ABC pour le nouveau nectar et calculer la position et la modération. Si le nouvel emplacement est meilleur que l'emplacement d'origine, remplacer l'emplacement d'origine.

Étape 4 Chaque processus compare le nombre de cycles d'itération avec maxcycle, si cycle> maxcycle, enregistrer la solution optimale; sinon passer à l'étape 3.

Étape 5 Chaque processus envoie les résultats des processus esclaves au processus principal, puis quitte l'environnement parallèle et fournit la meilleure solution finale.

Levy flight ABC :

Bien qu'ABC soit plus apte à établir un équilibre entre exploration et exploitation par rapport à d'autres algorithmes, celui-ci présente plusieurs lacunes [14]. Bien qu'il puisse mieux effectuer la recherche globale en effectuant des recherches aléatoires autour de chaque source de nourriture et en couvrant au mieux l'espace de recherche, il rencontre plusieurs problèmes au niveau de l'exploitation et reste parfois bloqué sur des minimums locaux, en particulier dans les fonctions multimodales complexes [14].

Si l'algorithme ABC d'origine ne parvient pas à améliorer la solution pour une source de nourriture particulière pour un nombre déterminé d'itérations (paramètre : limit), alors l'abeille employée ayant cette source devient une abeille scoute. Celle-ci, alors est affectée de manière aléatoire dans l'espace de recherche, dont les limites sont déterminées. Dans ce cas, les abeilles scoutes sélectionnent une zone de l'espace de recherche de manière complètement aléatoire, alors qu'il serait judicieux d'utiliser les résultats obtenus jusqu'à ce moment-là grâce à la succession de cycles de l'algorithme. Il est très peu probable que cette source de nourriture récemment sélectionnée soit supérieure à la meilleure source de nourriture jusque-là. Les abeilles scoutes trouvent de nouvelles sources de nourriture en utilisant la distribution Levy Flight avec également la prise en compte de la meilleure solution réalisée au lieu de choisir une source de nourriture aléatoire. En effectuant suffisamment de recherches aléatoires, l'exploration de l'algorithme ABC, qui est déjà efficace, est améliorée. La réaffectation d'une solution à l'abeille scoute s'effectue selon l'équation suivante :

$$X^{t+1} = X^t + \alpha \oplus Lèvy(\beta)$$

α : Pas aléatoire.

ABC chaotique floue :

Un algorithme ABC chaotique basé sur les système flous a été proposé dans [15]. La logique floue est utilisée pour : l'élimination de l'ambiguïté pendant que le chaos est utilisé pour générer une meilleure diversité dans la population initiale de l'algorithme d'optimisation des colonies d'abeilles

La théorie du chaos est introduite pour trois principales raisons :

• Diversité croissante dans la population initiale.

• Trouver le voisinage autour d'une source de nourriture.

• Générer des nombres aléatoires

La décision d'affecter une abeille à l'une des trois classes, à savoir scoutes, employées et spectatrices, est basée sur la fonction d'appartenance floue avec un seul paramètre: la condition physique.

Une base de règles gaussienne avec une fonction d'appartenance gaussienne a été définie manuellement et utilisée pour la classification des abeilles. De plus, l'optimisation ABC standard basée sur la carte chaotique et la classification d'abeille floue est utilisée pour l'optimisation.

1.4 Hybridations de l'algorithme des abeilles artificielles

De nombreuses hybridations sont appliquées à l'algorithme des abeilles à l'aide d'autres techniques d'optimisation afin d'améliorer sa performance, dans le tableau suivant on présente les motifs ayant conduit à de telles hybridations **[3]**:

Technique d'hybridation	Objectif	Référence
Méthode des moindres carrés	Améliorer la vitesse de convergence en utilisant les moindres carrés pour optimiser les portions linéaires de l'espace de recherche.	[19]
Evolution différentielles	Prévient la convergence rapide des évolutions différentielles et leur stagnation dans un optimum local.	[20]
Evolution quantique	Améliore la vitesse de convergence et	[21]

	augmente la diversité de la population	
Optimisation par les essaims particulaires.	Améliore le taux de convergence de l'algorithme hybride tandis que la faculté exploratoire de l'ABC améliore la recherche dans l'espace de recherche	[22]
Levenberq-Marquardt	Exploite la convergence rapide de Levenberq-Marquardt pour initialiser l'ABC afin d'approcher la solution optimale	[23]
Optimisation par colonie de Fourmies (ACO)	Exploite la simplicité et la robustesse de l'ACO. La capacité de recherche de l'algorithme ABC est utilisée pour échapper au minimum local. L'algorithme hybride est capable de surmonter les inconvénients de l'ACO qui ne convient pas aux optimisations continues	[24]

Tableau 7: Hybridations de l'algorithme des abeilles

2. Algorithme des abeilles pour la recherche de l'information

Hormis le fait que de plus en plus de chercheur s'intéressent davantage à l'algorithme des abeilles, il a été jusque-là peu sollicité dans le domaine de la recherche d'information.

On peut citer :

2.1 L'optimisation par les essaims d'abeilles pour la recherche d'information

2.1.1 Approche BSO-IR

En [17] : l'algorithme des abeilles été adapté afin de trouver l'information recherchée par l'utilisateur, au sein d'une collection de documents assez volumineuse et ça a été considéré comme une contribution à la recherche web. De par leur taille immense, il n'est pas évident de rechercher dans ces collections de façon efficace en utilisant les approches de recherche d'information classiques par souci de performance en matière de vitesse de satisfaction des requêtes. Alors un algorithme basé sur l'optimisation par les essaims d'abeilles et baptisé : ''BSO-IR'' pour (Bees swarm optimization) a été conçu pour explorer des collection de documents dont le nombre est assez excessif, afin de trouver l'information dont l'utilisateur a besoin dans des délais concevables.

Mécanisme de recherche d'information adopté

Un système de recherche d'information retourne des documents dans un ordre sensé satisfaire le besoin d'un utilisateur exprimé par sa requête qui est composée de mots clés. Ceux-là sont traités et soumis à une fonction d'appariement vis-à-vis des documents de la collection afin d'en sélectionner les plus semblables à la requête. Les documents et la requête sont représentés grâce à un modèle et celui qui convient le mieux aux méta-heuristiques est le modèle vectoriel.

En [17] la mesure de similarité adoptée pour le modèle vectoriel est la similarité cosinus et la pondération utilisée pour le calcul des poids des termes de la requête et des documents est la pondération TF-IDF.

L'approche classique consiste à calculer la similarité entre la requête et tous les documents du corpus, cette approche possède une complexité de O(n*m) où n est le nombre de documents et m le nombre de termes or pour les corpus volumineux n et m, ont des ampleurs exponentielles. ''BSO-IR'' vient en alternative afin d'améliorer cette lacune toute en garantissant une qualité de sélection des documents similaire.

La méta-heuristique BSO

- Une abeille appelée **BeeInit** trouve une solution initiale présentant de bonnes caractéristiques. Celle-ci est appelée **Sref.**
- A partir de cette solution d'autres solutions de l'espace de recherche sont générées, l'ensemble de ces nouvelles solutions potentielles est appelé **SearchArea.** La taille de cette zone de recherche est le nombre d'abeilles, donc chaque solution de cette zone est attribuée à une abeille dont on confie la tâche de rechercher une meilleure solution que celle attribuée en la prenant comme point de départ.
- Chaque abeille communique les meilleurs résultats de ces recherches dans une structure appelée **Dance**.
- La meilleure solution dans **Dance** va remplacer la solution initiale **BeeInit** et va constituer le départ pour la constitution de la zone de recherche dans la prochaine itération.
- A chaque itération la nouvelle solution de référence Sref est stockée dans une liste Tabou afin de ne pas l'utiliser plus d'une fois.

Voici code 01 qui est le pseudo code de l'optimisation par les essaims d'abeilles (BSO)

Pseudo-code BSO :

Code 01 : Pseudo-code BSO

```
Début
Sref = solution trouvée par BeeInit ;
Tant que (MaxIter pas atteint ) faire
    Insérer Sref dans la liste tabou ;
    Déterminer la zone de recherche SearchArea à partir de Sref ;
     Assigner une solution de la zone de recherche à chaque abeille
     Pour i allant de 1 à k faire   //k étant le nombre d'abeilles
         Rechercher à partir de la solution assignée.
         Sauvegarder le résultat dans Dance.
     FP
     Calculer la nouvelle solution de référence Sref
FTQ
FIN
```

Algorithme BSO-IR pour la recherche d'information

Les étapes de cet algorithme sont conformes à l'algorithme BSO décrit ci-dessus donc comme suit :

1. **Phase d'initialisation :**

La solution de référence est construite à partir de termes inclus dans la requête via une heuristique.

2. **Phase de recherche :**

Un paramètre appelé **Flip** est introduit, ce dernier représente un pas. La génération d'une solution à partir de la solution de référence est réalisée en ajoutant ou supprimant des termes de cette solution. Ces termes sont situés dans le vocabulaire à des positions multiples du paramètre Flip.

En prenant l'un de ces termes, si celui-ci existe dans la solution en l'enlève sinon on l'ajoute.

Le code 02 ci-dessous représente l'algorithme utilisé dans cette stratégie:

Code 02 : stratégie de recherche n°1

Début

h = 0;

Tant que taille de la zone de recherche pas atteint et h< Flip **faire**

s = Sref;

p = 0;

Répéter

Si le terme à la position Flip*p+h existe dans s **alors** l'enlever.

Sinon l'ajouter à s ;

p = p+1;

Jusqu'à Flip*p+h ≥ n;

SearchArea = SearchArea ∪ {s};

h = h+1;

FTQ;

FIN.

Avec n : nombre de termes dans le vocabulaire.

K : nombre d'abeille qui est aussi la taille de la zone de recherche et le nombre de solutions potentielles.

Pour chacune des solutions trouvées, on calcul la similarité de celle-ci avec la requête et on sauvegarde le résultat dans la liste Danse.

Une seconde stratégie de recherche consiste à considérer Sref comme étant des paquets de termes contiguës, la solution s est générées en changeant le terme t_i du paquet s à partir de Sref

L'algorithme de cette stratégie est donné dans code 03 ci-dessous :

Code 03 : stratégie de recherche n°2

```
DEBUT
Tant que taille de la zone de recherche pas atteint et h< Flip faire
s = Sref;
p = 0;
répéter
  Si le  terme (k/Flip*h)+p existe dans  s alors l'enleverde s
  SINON insert it in s;
  p = p+1;
jusqu'à p >= k/Flip
SearchArea = SearchArea _ {s};
h := h+1;
FTQ;
FIN.
```

3. Détermination de la solution de départ pour la prochaine itération :

Si en matière de similarité, la meilleure solution dans l'itération en cours est supérieure à la solution de référence cette meilleure solution remplace la solution de référence et devient point de départ pour la génération de la zone de recherche pour la prochaine itération et le compteur d'essais permis est réinitialisé. Si Sref en cours, est toujours supérieure aux solutions trouvées par les abeilles au bout du nombre d'essais permis, une nouvelle solution de référence est choisie, celle-ci est la solution ayant le moins de termes en communs avec les solutions de références précédentes, cette solution est dite : meilleure en diversité.

Afin d'accélérer les performances de l'algorithme un algorithme de clustering à base de l'algorithme BSO est proposé afin de restreindre la recherche dans le cluster comprenant la solution de référence et ainsi optimiser le temps de réponse encore davantage.

Analyse et comparaisons :

L'algorithme BSO présente quelques différentes par rapport à l'algorithme de colonies d'abeilles artificielles à savoir :

- Pour BSO Le feedback positif a lieu grâce à l'adoption d'une nouvelle solution de départ pour l'ensemble des abeilles, ceci équivaut avoir une seule abeille employée et les reste c'est des abeilles spectatrices, dans le cas de l'algorithme ABC.
- L'exploration et l'exploitation se fait dans un seul site pour BSO, alors que des sites au nombre des abeilles employées sont explorés et exploités dans L'algorithme ABC.
- L'aspect convergence vers la solution optimale est moins évident dans BSO que dans l'ABC.

Toutefois ces différences ne signifient pas la supériorité d'une approche sur une autre mais plutôt que BSO soit un cas particulier de l'ABC ayant ses usages spécifiques.

Un autre point à souligner pour BSO-IR est que le fait d'ajouter ou modifier des termes à une solution modifie énormément son contenu ce qui est ambigu, car en sortie qu'elle serait la structure des documents retournés à l'utilisateur ?

2.1.2 Approche BSOGDM

En [18] l'auteur a repris l'optimisation BSO en lui associant un clustering à l'aide de l'algorithme K-means avec l'extraction des termes fréquents fermés pour chaque cluster ainsi une préparation et des connaissances supplémentaires sont apportées sur l'espace de recherche afin que la recherche par les abeilles soit plus intelligente et efficace et minimiser le compromis : temps de réponse et qualité de solution.

Conclusion

A travers ce chapitre l'état de l'art dans le domaine de cette étude a été lu scruté et analysé afin de comprendre la manière dont la communauté scientifique a abordé le sujet, ses trouvailles et ses recommandations.

Le chapitre a fait état de ce qu'a été réalisé et ce que ne l'est pas, et a offert une meilleure vision de la manière avec laquelle résoudre les questions soulevées et les différentes étapes pour ce faire.

Chapitre 03

Conception et implémentation de l'approche proposée

Introduction

Après avoir vus les différents travaux réalisés dans le domaine de la RI en utilisant l'algorithme des abeilles nous présentons notre démarche afin de répondre à la problématique et ce afin de garantir l'intégrité des documents manipulés.

Celle-ci consiste à réaliser une segmentation de l'espace de recherche des documents en regroupant les documents les plus similaires dans des clusters afin de créer des sites riches pour l'exploitation avant de faire appel aux abeilles afin de les explorer et les exploiter, en effet des documents similaires en un même endroit augmentent le feedback positif en cas de bonne fitness et inversement augmentent le feedback négatif en cas de fitness médiocre impliquant une meilleure convergence de l'algorithme des abeilles car on ne s'attarde pas donne une zone où il n'y a pas de bonnes solutions . Ainsi l'effort d'exploitation va être concentré sur une partition de l'ensemble des documents plus précisément ceux faisant partie des meilleurs clusters.

Ensuite on présente le prototype réalisé qui est un moteur de recherche puis nous allons lister les tests effectués les résultats obtenus et les conclusions à tirer.

1. Approche proposée : prétraitement et recherche

Afin d'apporter une solution au problème de pertinence de la recherche et délai d'attente de l'utilisateur dans la recherche d'information dans les grandes collections de documents en utilisant l'algorithme des abeilles, nous nous intéressons à la décomposition de ce temps de réponse.

L'algorithme des abeilles est un algorithme itératif comportant :

- Une phase d'initialisation à SN itérations qui est le nombre d'abeilles employées.
- Un deuxième bloc itératif à trois phases :
    - Phase des abeilles employées :SN itérations.
    - Phase des abeilles spectatrices : SN itérations.
    - Phase des abeilles scoutes : itérations au nombre des sources de nourriture épuisées

Le 2è bloc lui-même ayant MaxIter itérations qui est un paramètre prédéfini de l'algorithme, il se peut qu'on définisse une condition d'arrêt autre que le nombre max d'itérations, si cette condition n'est pas vérifiée l'algorithme va poursuivre ses itérations.

La bonne utilisation de ces deux paramètres MaxIter et SN décrits ci-dessus apporte énormément d'influence sur le temps de réponse. Or ça n'a pas de sens de réduire ses deux paramètres sans réaliser des résultats satisfaisant en matière de qualité de solution ou sans atteindre l'objectif derrière l'algorithme. Donc afin de réduire le temps de réponse il est primordial de réduire le nombre d'itérations nécessaires à l'obtention des résultats souhaités.

Pour ce faire une convergence rapide de l'algorithme vers un optimum s'avère incontournable.

On remarque comme dans l'exemple d'optimisation numérique, les zones de l'espace de recherche ayant des tendances positives ou négatives aident l'algorithme à converger grâce aux feedback positif et feedback négatif. c'est-à-dire que là où la courbe tend vers un minimum la probabilité de recrutement p_i vers la source de nourriture i des abeilles spectatrices augmente donc la probabilité de découvrir le minimum augmente en conséquence ce qui est l'objectif de l'algorithme ABC pour l'optimisation numériques. Là où la courbe est loin du minimum cette zone possède une tendance négative donc un feedback négatif a lieu et l'abeille quitte cette zone ce qui offre une probabilité que la solution qui va être générée aléatoirement pour cette abeille soit dans un voisinage profitable.

Dans le cas des collections de documents, la disposition de ces documents est généralement aléatoire, donc il n'y a pas vraiment de tendance régionale, car des documents successifs dans leur ordre ne partagent pas forcément le même sujet ou thématique.

Ainsi, un prétraitement de la collection à utiliser pour la recherche est susceptible de provoquer une meilleure convergence vers les solutions. Si la disposition des documents dans la collection est trop aléatoire et qu'elle est laissée telle qu'elle il se peut qu'il n'y ait jamais de convergence.

Dans ce qui suit nous allons proposer deux techniques pour pallier au problème des collections dont les documents sont très aléatoires.

1.1 Prétraitements de la collection à utiliser

1.1.1 Substitution du document dans la prochaine position par le voisin le plus proche:

Cette technique consiste à modifier la disposition des documents dans la collection de sorte à avoir des documents successifs qui se ressemblent. Ainsi pour le premier document on calcule la similarité avec tous les autres documents du corpus. Ainsi le document ayant le meilleur score de similarité va être placé en 2^{è} position dans le corpus. Pour le document qui vient d'être placé dans la position n°2 on refait la même opération en partant de la 3è position dans les comparaisons. L'opération est répétée jusqu'à réordonner tous le corpus. Si jamais pour un document les scores de similarité sont tous nuls pour les documents dans des positions qui lui sont inférieures l'ordre est maintenu et on passe au document suivant. Le code 03 présente l'algorithme de repositionnement des documents :

code 03 : Repositionnement des documents

```
Début
tc = taille de la collection;
meilleure_sim = 0 ;
Pour i allant de 1 à tc faire
   pour j allant de i+1 à tc faire
       Sim = Calculer_similarité(doc[i],doc[j]);
       Si (sim > meilleure_sim) alors
            position = j;
       FSI
       j = j+1;
   FP
Permuter (doc[i+1], doc[position]);
i = i+1;
FP
FIN
```

Bien que cette technique soit coûteuse en terme de calculs car la complexité est en O(tc^2), elle reste envisageable dans le cas où ces calculs sont distribués, ou bien pour de supers calculateurs.

1.1.2 Faire un clustering avec l'algorithme des abeilles artificielles:

Initialement les documents du corpus sont affectés au cluster 0, ils seront au fur et à mesure regroupés dans des clusters à base de leur similarité. Ce programme est conçu pour s'exécuter comme tâche planifiée ou dans les moments d'inactivité du système, il peut aussi fonctionner en arrière-plan. Si le programme s'arrête prématurément ou s'il est arrêté par l'utilisateur, à la prochaine exécution il doit poursuivre là où il s'est arrêté.

Pour effectuer ce clustering cet algorithme commence du premier document du corpus lors du premier démarrage du programme, sinon du dernier document en cours. Le document en cours est considéré comme le document de référence du cluster qui va être créé à la base des documents qui lui sont similaires et ce document de référence va se distinguer des autres grâce à sa similarité égale à 1. On répartit ensuite les abeilles employées, uniformément sur le corpus. Chaque abeille recherche dans son voisinage limité par des bornes supérieures et inférieures pour trouver des documents ayant une similarité avec le document susmentionné supérieure au paramètre **sim_min**. le programme crée un cluster et y affecte les documents qui sont similaires au document en question et sauvegarde aussi la similarité trouvée entre le document de références et ses documents associés. On affecte ensuite les abeilles spectatrices selon la probabilité calculée vers les voisinages des abeilles employées, ces spectatrices vont rechercher chacune dans son voisinage une valeur supérieure à sim_min. A chaque fois qu'une bonne solution est découverte dans son voisinage l'abeille employée migre vers la nouvelle solution.

Dans cette recherche on peut tomber sur un document déjà affecté à un cluster, si la similarité avec le document de référence du cluster en cours est supérieure à la similarité avec celui du cluster d'affectation, le cluster en cours récupère le document en question car il lui est plus similaire.

Si jamais pas d'amélioration des solutions pour une abeille employée après un nombre maximum d'itérations on lui génère une nouvelle solution aléatoirement.

A chaque cycle on répète ce même procédé avec un autre document de référence généré aléatoirement parmi les documents qui ne sont pas encore affectés à un cluster.

Voici en Code 04 le Pseudocode de la méthode et en Code 05 une procédure qui en découle :

Code 04 : Algorithme BeeClustering

Début

TC = taille_collection ;

SN = Nombre_solutions ; \\nombre d'abeilles employées

NAS = nombre d'abeilles spectatrices ;

x = position dernier_doc_en_cours;

IterMax= paramètre_IterMax ;

Nb_iter = 0 ;

Tant que des documents sont affectés au cluster 0 **faire**

Pour i = 1 à SN **faire**

Position_abeille[i] =round($\frac{\mathbf{TC}}{\mathbf{SN}} \times (\mathbf{i} - \frac{\mathbf{I}}{\mathbf{2}})$ **)**;

Fitness_abeille[i] = SIM(doc[x], doc[Position_abeille[i]]) ;

Evaluer(Fitness_abeille[i]) ;

FP

Nb_iter = 0 ;

Répéter

Pour i = 1 à SN **faire**

Voisinage_abeille[i] = Position_abeille[i] +round($\frac{\mathbf{TC}}{\mathbf{2SN}} \times \mathbf{rand}[-\mathbf{1}, \mathbf{1}]$);

Fitness__voisinage_abeille[i] = SIM(doc[x],doc[Voisinage_abeille[i]]) ;

Si Fitness__voisinage_abeille[i] > Fitness_abeille[i] **alors**

Position_abeille[i] = Voisinage_abeille[i] ;

Fitness_abeille[i] = Fitness__voisinage_abeille[i] ;

Fsi

Somme_fitnesses = Somme_fitnesses + Fitness_abeille[i]

Evaluer(Fitness_abeille[i]) ;

FP

Pour i = 1 à SN **faire**

Proba[i] = $\frac{\textbf{fitnesse\_abeille[i]}}{\textbf{Somme\_Fitnesses}}$

Nb_abeilles_spec_source_i = round(NAS x Proba[i]) ;

Pour j = 1 à nb_abeilles_affectee_a_source_i **faire**

Voisinage_abeille[j]=position_abeille[i]+round($\frac{TC}{2SN} \times \boldsymbol{rand}[-\mathbf{1}, \mathbf{1}]$);

Fitness_voisinage_abeille[j] = SIM(doc[x],doc[Voisinage_abeille[j]]) ;

Si Fitness__voisinage_abeille[i] > Fitness_abeille[i] **alors**

Position_abeille[i] = Voisinage_abeille[i] ;

Fitness_abeille[i] = Fitness__voisinage_abeille[i] ;

Fsi

Evaluer(Fitness_abeille[j]) ;

FP

FP

Réinistialiser_aléatoirement_abeilles_scouts() ;

Nb_iter ++ ;

Jusqu'à nb_iter = IterMax -1 ;

x= prochain_doc_à_traiter() ;

FTQ

FIN

SIM est la fonctionn qui calcule la similarité cosinus entre 2 documents

Code 05: Procédure : Evaluer(Fitness_abeilles[i])

```
Début
Si Fitness_abeilles[i] > Fitness_min alors
   Si doc[position_abeille[i]] déjà affecté alors
      Si sim(doc[position_abeille[i]], doc[x])> sim (doc[position_abeille[i]], doc[aff])
            Retirer doc[position_abeille[i]] du cluster auquel il est affecté;
      Fsi
      Insérer position_abeille[i] dans le cluster C du document x ;
   Fsi
Sinon
     IterAbandon_abeille_i ++ ;
Fsinon
```

1.2 Algorithme de recherche d'informations BeeClustSearch

Peu importe la méthode choisie pour le prétraitement de la collection, l'algorithme de recherche **BeeClustSearch** va explorer l'espace de recherche afin de trouver les meilleurs documents répondant à une requête. Dans le cas du prétraitement par substitution de la prochaine position par le plus proche voisin, il n'y a pas de groupement en clusters, on peut donc considérer les documents comme faisant tous partie du même cluster C[0] et y affecter toutes les abeilles. Cette idée peut même être appliquée à la collection sans aucun prétraitement même si ce n'est fortement pas recommandé. Au démarrage, le programme consulte la liste des clusters disponibles. Si le nombre d'abeilles est supérieur au nombre de clusters on les affecte équitablement sur les clusters. Sinon on affecte les abeilles aux SN premiers clusters les plus volumineux. La recherche est alors effectuée dans chacun des clusters par l'abeille employé qui y est affectée. A chaque itération, dans la phase des abeilles spectatrices, celles-ci sont recrutées par le biais de la probabilité p_i qui mesure l'apport de chaque cluster en terme de fitness par rapport à l'ensemble des autres clusters. Ainsi le cluster ayant le plus d'apport se voit attribué plus d'abeilles pour concentrer plus d'effort dans son exploitation.

Les clusters avec les apports les plus faibles doivent rapidement être abandonnés et ainsi les abeilles employées précédemment affectées à ces clusters migrent vers des clusters dont on a

pas affecté d'abeilles jusque-là. Ainsi au fur et à mesure de la progression de l'algorithme dans la recherche, de plus en plus de clusters vont être explorés et exploités couvrant ainsi la totalité du corpus.

Voici en Code 06 l'algorithme de recherche BeeClustSearch :

Code 06 : l'algorithme de recherche BeeClustSearch

Début

TC = taille_collection ;

SN = Nombre_solutions ; \\nombre d'abeilles employées

NAS = nombre d'abeilles spectatrices ;

Query = Requête de l'utilisateur ;

//début phase d'initialiastion

Tant que i =< SN **faire**

Si SN < nb_clusters **alors**

Choisir SN cluster ;

Sinon

Choisir tous les clusters

Fsinon

Pour chaque cluster **faire**

Nb_abeilles_clus = $\frac{SN}{NB\_clusters}$

Sélection de nb_abeille_clus positions aléatoirement dans le cluster en cours et mettre le résultats dans resultat_selection;

J = 0 ;

Tant que j < nb_abeilles_clus **faire**

Position_abeilles[i] = resultat_selection [j] ;

Fitness_abeille[i] = SIM(Query, doc[Position_abeille[i]]) ;

Evaluer (Fitness_abeille[i]) ;

i++ ;

```
            j++ ;
        FTQ
    FP
FTQ
//fin phase d'initialisation
Répéter
  //phase abeilles amployées
    Pour i = 1  à SN faire
        Voisinage_abeille[i] = position d'un document du même cluster que
                               Position_abeille[i];
        Fitness__voisinage_abeille[i] = SIM(Query,doc[Voisinage_abeille[i]]) ;
        Si Fitness__voisinage_abeille[i] > Fitness_abeille[i] alors
            Position_abeille[i] = Voisinage_abeille[i] ;
            Fitness_abeille[i] = Fitness__voisinage_abeille[i] ;
            Evaluer(Fitness_abeille[i]) ;
        Sinon
            Nb_iter_abandon_abeille[i]++ ;
        Fsinon
        Somme_fitnesses = Somme_fitnesses + Fitness_abeille[i] ;
    FP

    Pour i = 1  à SN faire
        Proba[i] = fitnesse_abeille[i] / SommeFitnesses
        Nb_abeilles_spec_source_i = round(NAS x Proba[i]) ;
        Pour j = 1 à nb_abeilles_affectee_a_cluster_i faire
            Voisinage_abeille[j]=position dans le même cluster que
            position_abeille[i] ;
```

```
                Fitness_voisinage_abeille[j] = SIM(Query,doc[Voisinage_abeille[j]]) ;
                Si Fitness__voisinage_abeille[i] > Fitness_abeille[i] alors
                    Position_abeille[i] = Voisinage_abeille[i] ;
                    Fitness_abeille[i] = Fitness__voisinage_abeille[i] ;
                    Evaluer(Fitness_abeille[j]) ;
                Sinon
                  Nb_iter_abandon_abeille[i]++ ;
                Fsinon
            FP
        FP
                Réinistialiser_aléatoirement_abeilles_scouts() ;
Jusqu'à nb_itérations_max ;
FIN
```

Code 07 : Procédure : Evaluer(Fitness_abeilles[i])

```
Début
        Si liste_resultats.taille < nombre_docs_a_retenir alors
            Ajouter position_abeille[i] ;
        Sinon
            Si fitness_abeille[i]>liste_resultats[0].fitness alors
                Ajouter (position_abeille[i]) ;
                Supprimer(liste_resultats[0]) ;
                Liste_resultats.trier ();
        Fsinon
```

2. Implémentation de l'approche : prétraitement et recherche

2.1 Choix des techniques à réaliser

Comme décrit au début de ce chapitre, l'utilisation de l'algorithme des abeilles pour la recherche d'information comme le stipule l'approche proposée nécessite au préalable un prétraitement du corpus à utiliser afin d'optimiser le temps de recherche, pour cela deux approches ont été proposées à savoir : la substitution du prochain document par le voisin le plus proche du document en cours et le clustering avec les abeilles or la première méthode est coûteuse en temps de traitement pour le corpus tout entier, celui-ci est certes pas trop important par rapport au temps d'exécution lors de la recherche mais risque de ne pas se terminer rapidement afin de permettre de tirer de bonnes conclusions quant à son utilité. Dès lors l'approche de clustering avec les abeilles a eu la priorité pour l'implémentation.

Aussi pour des fins de comparaison une approche classique utilisant le modèle vectoriel et la similarité cosinus pour la recherche d'information a été choisie afin de la confronter à la recherche par les abeilles. Cette approche classique calcule la similarité entre la requête et tous les documents de la collection et retourne un classement décroissant des documents par ordre de leur similarité avec la requête.

Modèle de recherche d'information choisi : Le modèle d'appariement entre les requêtes et les documents est le modèle vectoriel car selon **[17]** c'est le plus approprié aux méta-heuristiques.

Algorithme de stemming : l'algorithme de stemming intégré à l'application est l'algorithme de Lancaster, le choix est fait en correspondance avec le corpus utilisé dont la liste des tokens est traitée avec ce stemmer.

2.2 Présentation du corpus utilisé

Le corpus utilisé : Reuters Corpus Volume 1 (RCV1) est une archive de plus de 800000 articles de presse appartenant à l'agence de presse internationale Reuters. Ces documents concernent la période entre le 10 août 1996 au 19 août 1997. Cette archive possède 2 versions la deuxième apporte des corrections à la première. Elle contient 804414 documents indexés de 2286 à 810935.Chaque document possède un identifiant. Les documents sont disposés par ordre de cet identifiant majoritairement hormis quelques exceptions où toute une séquence de documents est déplacée.

Elle possède 5 fichiers index dont les lignes ont la structure suivante :

id_document␣␣id_term_i:w_i␣id_term_j:w_i.....etc .

Ces cinq fichiers ont été fusionnés en un seul index.

Où w_i est le poids du terme i dans ce document selon le schéma de pondération TF-IDF. Ces poids sont déjà normalisés cosinus. c'est-à-dire multipliés par : $\frac{1}{\sqrt{\sum_{i=1}^{m} wi^2}}$.

m = nombre de termes dans le corpus.

Les nombre de termes dans cette collection est 47235 termes ceux-là sont normalisés et le steming de Lancaster leur a été appliqué , ils sont présentés comme suit :

terme␣id_term␣idf.

2.3 Autres choix techniques

La base de données :

Une base de données assez simple sous le SGBD mysql est conçue pour le suivi de l'opération de clustering et l'utilisation des clusters. A la recherche d'un SGBD à la fois léger et performant j'ai testé les requêtes d'insertion et surtout de mise à jour car c'est la manipulation la plus fréquente dans le programme, sur les SGBD : Firebrid et Mysql et il s'est avéré que mysql surpasse Firebird en temps d'exécution des différentes requêtes.

La base de donnée possèd les tables suivantes :

Table	Champ	Type de champ	Représentation
Docs	Doc_id	Entier	Identifiant du document
	Cluster	Entier	N° de cluster d'affectation
	fitness	Réel	Similarité entre ce document et le document de référence du cluster
Etat	Doc_en_cours	Entier	Document de référence en cours de traitement
	Cluster_en_cours		Cluster en cours de construction
Termes	Term_id	Entier	Identifiant du terme
	Term_text	Chaine de caractères	Texte du terme
	idf	Réel	Inverse fréquence documentaire

			du terme

Tableau 8: Structure base de donnée de suivi de clustering

La table docs contient la liste des identifiants des documents du corpus, elle est utilisée pour l'affectation des documents aux clusters correspondants grâce à leur fitness. Celle-ci est stockée dans le champ fitness et le cluster d'affectation est renseigné avec le champ cluster.

La table état contient le document de référence en cours de traitement afin d'y associer les documents similaires au même cluster.

L'environnement de développement :

L'application est réalisée dans un environnement .net sous l'IDE : MS Visual studio 2010. avec le langage de programmation c#. Le framework utilisé est le .net Framework 4.0

2.4 Présentation de l'application réalisée

Le prototype réalisé est un **moteur de recherche** dans le corpus RCV1, il comprend deux méthodes de recherche une méthode classique qui calcule la similarité entre la requête saisie par l'utilisateur dans le masque de recherche avec tous les documents du corpus et retourne une liste ordonnée des documents par ordre croissant du score réalisé, les scores nuls ne sont pas retenus.

Il inclut aussi la recherche en utilisant l'algorithme des abeilles dans ce même corpus, pour chacune des deux méthodes le programme affiche le temps de réponse réalisé. Ce temps de réponse est enregistré grâce à la classe .Net prédéfinie : System.Diagnostics.Stopwatch, elle possède des méthodes comme start() pour démarrer le chronomètre, stop() pour l'arrêter et restart() pour le redémarrage. Tout au long de la programmation cette classe a été fréquemment sollicitée pour mesurer le coût de plusieurs instructions et boucles et détecter les pertes de temps afin de les éviter. On peut citer les accès répétitifs au disque qui consomment énormément à l'échelle de plusieurs itérations successives, parfois des volumes importants de données sont chargés en mémoire une fois pour toute car nécessitent des accès très fréquents comme dans le cas de l'index comportant les documents du corpus, celui-ci occupe une capacité de 1,4 Go sur le disque. Et porte la consommation de l'application à 3Go de RAM une fois chargé en mémoire.

Voici en figure 6 : une prise d'écran du prototype réalisé.

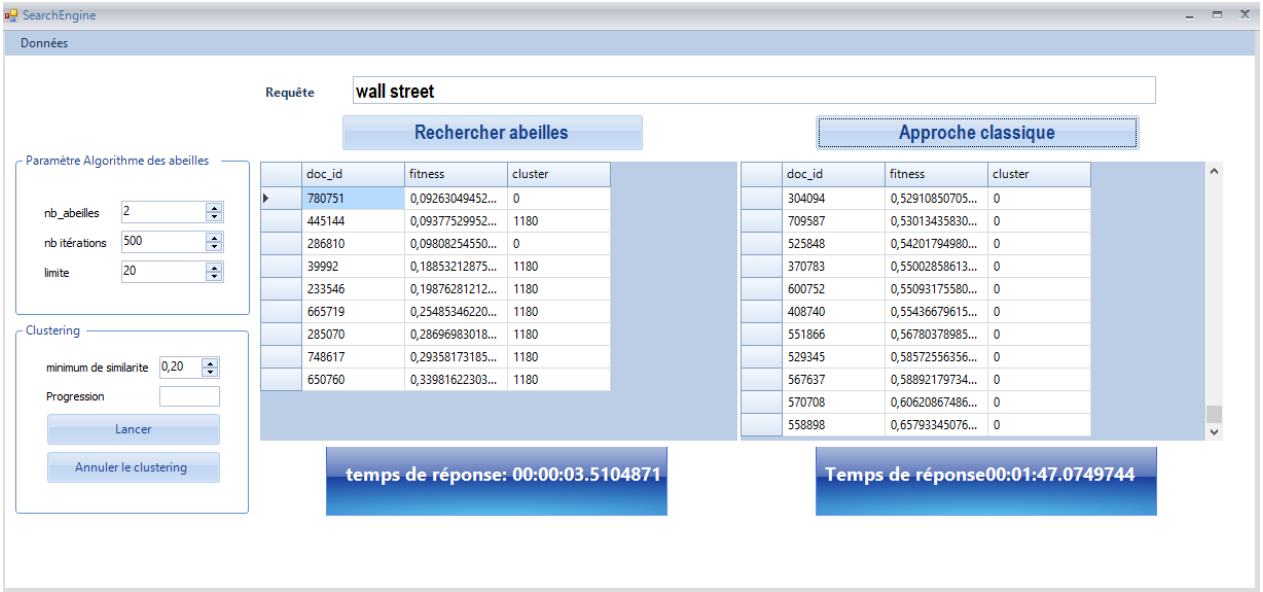


Figure 6 prise d'écran de l'application réalisée

3. Expérimentations et résultats

Les expérimentations ont été menées sur un ordinateur personnel avec processeur Intel Core2 Duo avec une fréquence de : 2.10 GHz et une mémoire vive de 6GO.

3.1 Test de l'algorithme de recherche BeeClustSearch

Requête 1: « oil price records »

Algorithme	Nombre d'abeilles	Nombre d'itérations	limite	Similarité obtenue	Temps de réponse
ABC	50	50	10	0.244	35 sec
ABC	30	50	10	0.2941	22 sec
ABC	10	50	10	0.1937	6 sec
ABC	1	50000	10	0.2055	1mn 30 sec
ABC	1	10000	10	0.200033	26 sec

Approche Classique	Id du Meilleur document : 345199	0.40815	1 mn 26 sec

Tableau 9 tests pour la requête1

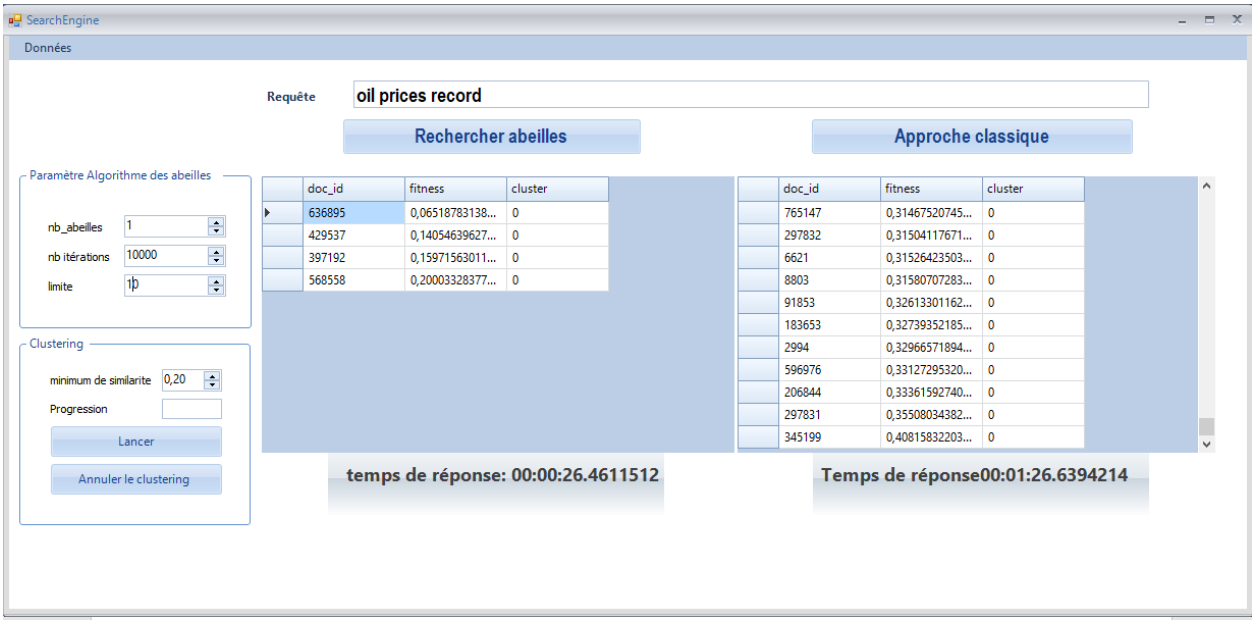


Figure 7 illustration requête 1

Requête 2 : « stock income market»

Algorithme	Nombre d'abeilles	Nombre d'itérations	limite	Similarité obtenue	Temps de réponse
ABC	1	1000	10	0.1958	2 sec
ABC	1	10000	10	0.2314	20 sec
ABC	10	50	10	0.23112	7 sec
ABC	1	50000	10	0.2304	1mn 41 sec
ABC	1	10000	10	0.22	23 sec
Approche Classique	Id du Meilleur document : 497567			0.4411	1 mn 36 sec

Tableau 10 tests pour la requête2

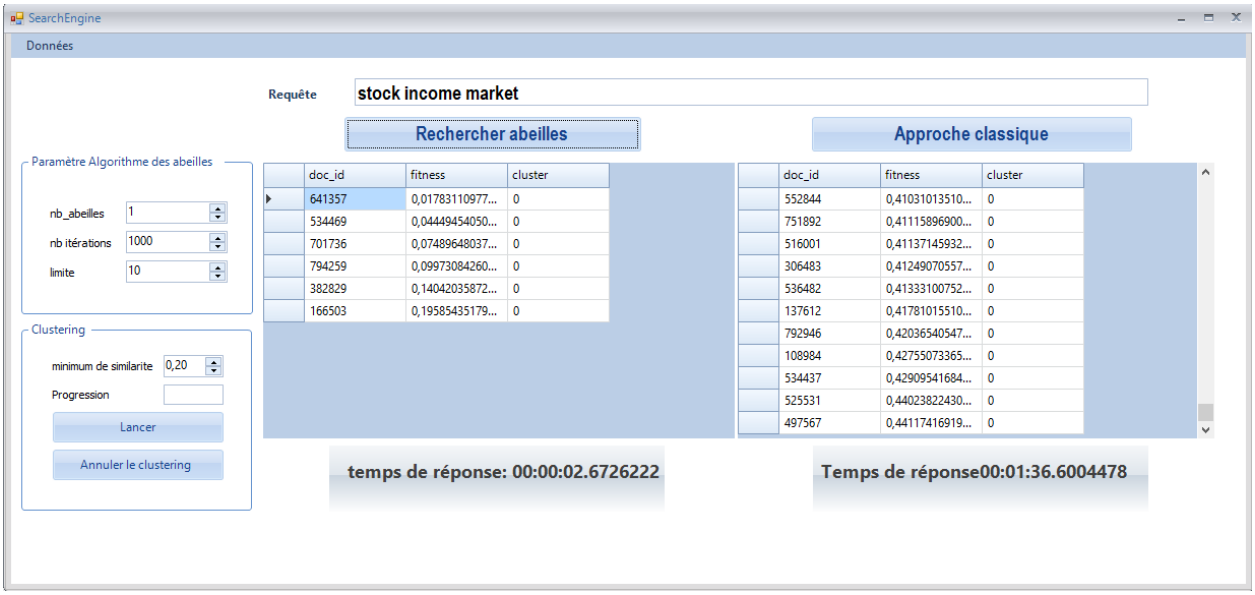


Figure 8 illustration requête2

Requête 3 : « information retrieval »

Algorithme	Nombre d'abeilles	Nombre d'itérations	limite	Similarité obtenue	Temps de réponse
ABC	30	20	10	0.2748	21 sec
ABC	1	10000	10	0.2319	20 sec
ABC	10	50	10	0.14	7 sec
ABC	1	50000	10	0.2304	1mn 35 sec
ABC	1	20000	10	0.21	38 sec
Approche Classique	Id du Meilleur document : 174587			0.455	1 mn 21 sec

Tableau 11 tests pour la requête3

Requête 4 : « artificial intelligence »

Algorithme	Nombre d'abeilles	Nombre d'itérations	limite	Similarité obtenue	Temps de réponse
ABC	20	30		0.3738	15 sec
Approche Classique	Id du Meilleur document : 236589			0.54	1 mn 20 sec

Tableau 12 Tableau 1tests pour la requête 4

Figure 9 Illustration requête 3

3.2 Test de l'algorithme BeeClustSearch indépendamment de l'algorithme de clustering

Ce test part de l'hypothèse qu'on possède un algorithme de clustering puissant capable de placer les documents les plus similaires ensemble dans un même cluster, le but est de voir est ce que l'algorithme BeeClustSearch possède la faculté de converger vers une bonne solution.

Ainsi les six premiers résultats en matière de similarité obtenus par l'approche exacte (classique) en réponse à la requête : « barcelone atletico madrid » sont mis ensemble dans un cluster parmi les 30 premiers cluster en matière d'effectifs, peu importe quel cluster, pour ce test on a choisi le 29^{è} plus grand cluster dont le numéro est : 1203.(on rappelle que cet algorithme de recherche commence par les clusters les plus volumineux en cas de nombre de cluster supérieur au nombre d'abeilles utilisées), ce cluster contient 286 documents soit un bon document pour 47 autres documents lorsqu'on affecte les six documents ayant mieux répondu à la requête. Donc ce

cluster est considéré comme un site riche pour l'algorithme des abeilles. Les résultats sont comme suit :

Algorithme	Nombre d'abeilles	Nombre d'itérations	limite	Similarité obtenue	Temps de réponse	Classement de la solution optimale
ABC	30	100	10	0.44	6 sec	5è
ABC	30	300	10	0.56	34 sec	2è
ABC	29	500	10	0.62	43 sec	1er
Approche Classique	Meilleur document : 576475			0.62	1 mn 14 sec	1er

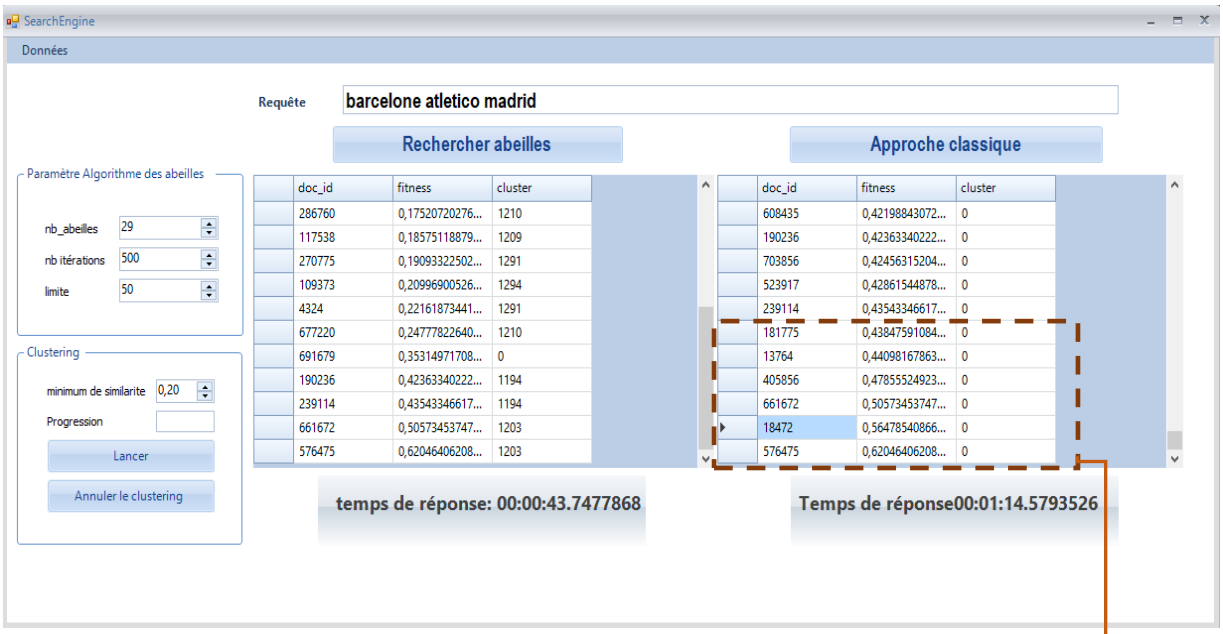


Figure 10 test Indépendant du clustering

3.3 Tests de l'algorithme BeeClustering

Configurations de paramètres	Nombre d'abeilles	Nb itérations	Fitness Minimum (similarité)	Temps de traitement d'un document	Nb moyen de documents dans un cluster
config 1	1000	7	0.1	31.97 sec	76
config 2	1000	7	0.5	16 sec	2
config 3	1000	25	0.5	1mn 2 sec	2
config 4	500	10	0.3	20	31
config 5	500	50	0.3	1mn 25 sec	29
config 6	1000	10	0.3	35 sec	6
config 7	10000	10	0.3	6 mn 42 sec	62

Tableau 13 Paramètres et résultats de clustering

4. Discussion

- Les tests réalisés indépendamment de l'algorithme de clustering proposé sont très prometteurs et montrent que l'algorithme BeeClustSearch en l'existence d'une bonne préparation du corpus converge rapidement vers une bonne solution, car en rencontrant une seule bonne solution dans un cluster, des abeilles spectatrices vont être recrutées vers ce cluster et au fur et à mesure de la progression des itérations l'abeille employée va se déplacer vers la meilleure solution trouvée.
- Les temps de réponse obtenus de recherche par les abeilles sont bons voire significativement meilleurs dans certains cas par rapport à ceux de l'approche classique, même si la qualité des solutions reste à améliorer.

- La similarité obtenue par l'algorithme classique est la meilleure étant donné que cette approche est exacte car elle parcourt tous l'espace de recherche. Par contre l'approche par l'ABC qui est une heuristique n'explore pas tout l'espace de recherche.
- Le programme de clustering s'est avéré couteux dans les calculs, plusieurs révisions et mises à jours lui ont été apportées réduisant un peu son temps de traitement. A l'heure de l'établissement de ces tests le nombre de document déjà affectés s'élève à seulement 57837 documents soit près de 7% du corpus ce qui implique que les jugements faits ne sont pas définitifs.
- Le paramétrage contribue largement dans les qualités de solutions trouvées. Pour la recherche un équilibre entre le nombre d'abeilles et le nombre d'itérations permet d'obtenir de bons résultats et ce en matière de temps de réponse et de la qualité des solutions. Pour le paramétrage de l'algorithme de clustering un nombre d'abeilles important est nécessaire pour bien explorer l'espace de recherche qui est plutôt aléatoire (pas beaucoup de ressemblances pour les documents voisins).
- Pour le clustering le paramètre fitness_min fixé à 0.5 est très élevé car une population de 2 documents par cluster est inutile pour la recherche et intolérable.
- Avec 500 abeilles et 10 itérations on a 31 documents par cluster pour une fitness_min à 0.3 réalisé en 20 secondes semble la meilleure configuration même si un nombre un peu plus élevé de documents par cluster est souhaitable.

Conclusion

Dans ce dernier chapitre j'ai proposé une approche de préparation de la collection de documents à utiliser puis un algorithme de recherche par les colonies d'abeilles artificielles pour résoudre le problème de temps de réponse long dans l'interrogation de grands corpus de documents. A ce fait un moteur de recherche est réalisé permettant d'effectuer les expérimentations nécessaires. Il s'est avéré que le programme de clustering de la collection est tout de même couteux en temps d'exécution et n'a permis de classer qu'une infime partie du corpus, ne permettant pas de tirer toutes les conclusions. Toutefois les résultats sont prometteur surtout pour le temps de réponse qui peut avec un bon paramétrage de l'algorithme des abeilles artificielles être significativement supérieur à l'approche classique vue dans ce chapitre, la qualité des solutions quant à elle est très dépendante de la qualité du prétraitement appliqué à la collection de documents, en effet , si on arrive à regrouper les meilleurs documents répondant à une requête dans les mêmes clusters, la convergence de l'algorithme devient une évidence et en conséquence un meilleur temps de réponse car on pourrait atteindre un bon résultat par un minimum d'itérations.

Conclusion générale et perspectives

L'un des domaines d'intelligence artificielle les plus dynamiques et prometteurs est celui des algorithmes bio-inspirés tels que les algorithmes inspirés de sociétés d'animaux et d'insectes regroupés autour d'une finalité commune. L'un de ces algorithmes est l'algorithme de colonies d'abeilles artificielles.

A travers cette étude on s'adresse à la problématique de la lenteur des systèmes de recherche d'information pour répondre aux requêtes des utilisateurs, lorsqu'il s'agit de volumes de données immenses, en exploitant les avantages et mécanismes régissant le comportement des abeilles artificielles, on a conçu une approche pour minimiser le coût d'interrogation des corpus volumineux et le malaise que cela crée à l'utilisateur. On a procédé en deux phases :

- Préparation de la collection de données soit en changeant la disposition de ses documents ou par un regroupement de ceux-là grâce à leur similarité.
- Privilégier dans le processus de recherche les voisinages ou les clusters contribuant le plus à la qualité des documents retournés.

Ainsi un moteur de recherche concrétisant cette approche est réalisé, et des tests ont été conduits afin de juger de l'efficacité de la solution apportée.

Les résultats obtenus sont encourageants pour le temps de réponse de mon approche par rapport à l'approche classique testée qui est un parcours séquentiel de tout le corpus et le classement de tous les documents par ordre de similarité avec la requête. Quant à la qualité des documents retournés l'approche classique est relativement meilleure compte tenu que c'est une approche exacte.

La qualité de ce système de recherche d'information avec l'algorithme des abeilles est très dépendante de la qualité de la préparation préalable de la collection, Plus le clustering et de bonne qualité plus l'algorithme converge rapidement et plus les résultats sont meilleurs en qualité des documents sélectionnés et en temps de récupération de ceux-là. Ouvrant ainsi un horizon de recherche et d'amélioration très prometteur.

La présente étude ne constitue qu'une introduction à l'optimisation des systèmes de recherches d'information par des approches bio-inspirées combinées à des techniques de datamining. De ce fait quelques suggestions viennent naturellement à l'esprit ainsi on peut citer en guise de perspectives les points suivants:

- Proposer une version distribuée de l'algorithme des abeilles afin de réduire principalement les coûts de calcul relatifs à la préparation de la collection et aussi pour réduire considérablement le temps de réponse grâce à une convergence rapide de l'algorithme et grâce au parallélisme lors de la recherche.
- Utiliser d'autres approches d'apprentissage automatique en combinaison avec l'algorithme des abeilles afin de maximiser son apport.

Bibliographie

[1] J.hurwitz, D.kirsch,"Machine learning for dummies", John Wiley & Sons, Inc.2018

[2] T.M. Mitchell "Machine learning", McGraw-Hill 1997

[3] A.E. Hassanien, E.Emary, "Swarm Intelligence Principles, Advances, and Applications", CRC press 2016.

[4] W.Bruce Croft, D.Metzler , T.Strohman, "Search Engines Information Retrieval in Practice", Pearson Education 2010.

[5] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze "An Introduction To Information Retrieval", Cambridge University Press, 2009

[6] R. Baeza-Yates, B. Ribiero-Neto, "Modern Information Retrieval", Addison Wesley Longman Publishing Co. Inc.,1999.

[7] M.Zekri " Approches Bio-inspirées pour la Fouille de Données en Bioinformatique", thèse doctorat, université annaba 2015

[8] D. Karaboga. " An idea based on honey bee swarm for numerical optimization. Technical ReportTR06,ErciyesUniversity, 2005.

[9] D.Karaboga, B.Basturk (2007), "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems",12th International Fuzzy Systems Association congress.

[10] D.Karaboga,B.Basturk(2007)" A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", Journal of Global Optimization

[11] S.Shalev-Shwartz, S.Ben-David "Understanding Machine Learning:From Theory to Algorithms" , Cambridge University Press, 2014

[12] G.Yan, C.Li, "An effective refinement artificial bee colony optimization algorithm based on chaotic search and application for PID control tuning".Journal of Computational Information Systems, Vol. 7(9), pp. 3309-3316, 2011.

[13] Y.Hong, Z.Ji, , C.Liu, "Research of Parallel Artificial Bee Colony Algorithm Based on MPI"2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013), Atlantis Press, Paris, France, pp. 1352-1355.

[14] H.Hakli and H.Uguz, "Levy flight distribution for scout bee in artificial bee colony algorithm", Lecture Notes on Software Engineering, Vol. 1(3), 2013.

[15] V.Chahkandi, M.Yaghoobi, G.Veisi, “Feature selection with chaotic hybrid artificial bee colony algorithm based on fuzzy”, Journal of Soft Computing and Applications, pp.1-8, 2013.

[16] H. Drias, S. Sadeg, S. Yahi, “Cooperative Bees Swarm for Solving the Maximum Weighted Satisfiability Problem”, IWANN 2005: 318-325

[17] Drias, Habiba, and Hadia Mosteghanemi. "Bees swarm optimization based approach for web information retrieval." 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. Vol. 1. IEEE, 2010.

[18] Y.Djenouri, A.Belhadi, R.Belkebir “Bees swarm optimization guided by data mining techniques for document information retrieval” Expert Systems With Applications journal, 2018.

[19] A.Habbi ,Y.Boudouaoui, “Hybrid artificial bee colony and least squares method for rule-based systems learning” Thèse magistère, université de Boumerdès 2014

[20] J.Abraham, R.K.Jatoth, A.Rajasekhar, “Hybrid differential artificial bee colony algorithm” Journal of Computational and Theoretical Nanoscience,Vol. 9, pp. 1-9, 2012.

[21] H.Duan, C-F.Xu, Z-H.Xing, “A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problem”, International Journal of Neural Systems, Vol. 20(1), pp. 39-50, 2010.

[22] O.Altun T.Korkmaz : “Artificial Bee Colony Chain (PSOABCC): A Hybrid Metahueristic Algorithm” International Workshops on Electrical and Computer Engineering Koc University, 2014.

[23] H.Shah, R.Ghazali, N.M.Nawi, M.Deris, T.Herawan, “Global artificial bee colony Levenberq-Marquardt (GABCLM) algorithm for classification”, International Journal of Applied Evolutionary Computation, Vol.4(3), pp. 58-74, 2013.

[24] M. Kefayat, A. Lashkar Ara S.A. Nabavi Niaki, “A hybrid of ant colony optimization and artificial bee colony algorithm for probabilistic optimal placement and sizing of distributed energy resources” Energy Conversion and Management,Vol. 92, pp. 149-161, 2015.

[25] E. Bonabeau, M. Dorigo, G. Theraulaz, “Swarm Intelligence from natural to artificial systems”, Oxford University Press,1999.

Webographie

[W1] . http://www.tartarus.org/ ˜ martin/PorterStemmer/
[W2] . www.imerj.com
[W3] . www.scholarpedia.org/article/Artificial_bee_colony_algorithm.
[W4] . https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf
[W5] . https://www.idc.com/about