Université Saad DAHLAB - Blida 1

Faculté des sciences

Département d'Informatique

Mémoire présenté par :

Mlles MELILI Sara et AHMED AICHA Habiba

Pour l'obtention du diplôme de Master

Domaine : Mathématique et Informatique

Filière : Informatique

Tests et analyse d'algorithmes d'indexation sémantique dans le cadre de la proposition d'un système de recherche d'information sensible au contexte

Soutenu le : **21 octobre 2019**

Devant le jury composé de :

Mme OUKID Saliha	Présidente	Université de Blida 1
Mr FERFIRA Sofien	Examinatrice	Université de Blida 1
Mme. MEZZI Melyara	Promotrice	Université de Blida 1

Remerciement

Tout d’abord, nous tenons à remercier le bon Dieu le tout Puissant de nous avoir donné la force et le courage de mener à bien ce modeste travail, également nous remercions infiniment nos parents, qui nous encouragé et aidé à arriver à ce stade de notre formation.

Nous adressons nos remerciements les plus sincères à toutes les personnes qui nous ont permis d'évoluer dans la réflexion et l'élaboration de ce travail. Plus particulièrement, nous tenons à remercier :

Mme MEZZI, notre promotrice, que nous considérons comme une grande sœur ,vous nous avez beaucoup aide dans notre projet ,nous espérons réaliser tous ses rêves .

Mr.SIFI ,notre ami dans l’étude , nous voulons remercier cette personne à cause de lui, nous sommes engagés dans ce sujet

Enfin, nous tenons à remercier tous ceux qui nous ont aidés et assistés durant nos études et nous exprimons toute notre gratitude à vous.

Dédicace

Je dédie ce travail qui n'aura jamais pu voir le jour sans les soutiens indéfectibles et sans limite de mes chers parents qui ne cessent de me donner avec amour le nécessaire pour que je puisse arriver à ce que je suis aujourd'hui. Que dieux vous protège et que la réussite soit toujours à ma portée pour que je puisse vous combler le bonheur.

Je dédie aussi ce travail à mes deux frères Zakaria et Abdelkader et tout ma famille surtout oncle Redoune ce qui m'a ouvert ces portes pendant mes années d'étude

Je dédie aussi ce travail à mes amis proche surtout Linda,Ikram,yousra ,Habiba ,merieme,Hanane et Safaa

Je dédie aussi un ami très proche et près de mon coeur, j'ai souri ces trois dernière années à cause de sa présence dans ma vie. Bien que tu ne participeras pas le jour de remise des diplômes ,mais ton âme sera avec moi.

Dédicace

Résumé

La recherche d'informations textuelles sur Internet peut être l'application la plus utilisée en Recherche d'Information et cette application dérive des techniques de Traitement Automatique de la Langue et de l'indexation et consiste à récupérer une grande quantité d'enregistrements de texte. La récupération des informations est réalisée à l'aide d'outils informatiques appelés système de recherche d'informations (SRI).

Avec le développement Web, la recherche d'informations a fait face à de nouveaux défis d'accès à l'information et ce, pour trouver des informations pertinentes dans un espace diversifié et de grande taille. Dans ce contexte, nous proposons une méthode d'indexation contextuelle riche en sémantique afin d'obtenir des résultats de recherche satisfaisants et compatibles avec la demande de l'utilisateur.

A ce titre, une méthode d'indexation et d'appariement entre requête et documents est proposée et évaluées. Cette méthode se concentre sur les aspects sémantiques des documents et des requêtes avec un algorithme de stemming (racinisation) sémantiquement enrichi basé sur l'algorithme connu de Porter.

Mots clés : Recherche d'Information, Système de Recherche d'Information, Indexation sémantique, Sensibilité au Contexte.

Abstract

The search for textual information on the Internet can be the most used application in Information Retrieval (IR) and this application derives from techniques of Automatic Language Processing and Indexing and consists of recovering a large amount of text records. The retrieval of information is carried out using computer tools called Information Retrieval System (IRS). .

With web development, the search for information has faced new information access challenges to find relevant information in a large and diverse space. In this regard, we propose a contextual indexing method rich in semantics in order to obtain satisfactory search results that are compatible with the user's request.

As such, a method of indexing and matching queries and documents is proposed and evaluated. This method focuses on the semantic aspects of documents and queries with a semantically enriched stemming algorithm based on Porter's known algorithm.

Keywords : Information Retrieval, Information Retrieval System, Semantic Indexing, Context-sensitivity.

ملخص

يمكن أن يكون البحث عن المعلومات النصية على الإنترنت هو التطبيق الأكثر استخدامًا في استرجاع المعلومات (IR) ، ويستمد هذا التطبيق من تقنيات معالجة اللغة وفهرستها تلقائيًا ويكمن في استرداد كمية كبيرة من السجلات النصية. يتم استرجاع المعلومات باستخدام أدوات الكمبيوتر التي تسمى نظام استرجاع المعلومات .(IRS)
.

مع تطوير الويب، واجه البحث عن المعلومات تحديات جديدة في الوصول إلى المعلومات للعثور على المعلومات ذات الصلة في مساحة كبيرة ومتنوعة. في هذا الصدد، نقترح طريقة فهرسة للمحتوى غنية بالدلالات من أجل الحصول على نتائج بحث مرضية متوافقة مع طلب المستخدم.

على هذا النحو، تم اقتراح طريقة للفهرسة ومطابقة الاستفسارات والوثائق وتقييمها. تركز هذه الطريقة على الجوانب الدلالية للوثائق والاستعلامات باستخدام خوارزمية جذرية مخصبة بشكل يعتمد على خوارزمية بورتر المعروفة.

<u>الكلمات المفتاحية</u>: استرجاع المعلومات، نظام استرجاع المعلومات، الفهرسة الدلالية، حساسية السياق.

Table des matières

Liste des Figures

Liste des tableaux

Introduction générale

Introduction générale

1. Contexte et Motivations

Dans ce chapitre, nous parlons des algorithmes que nous avons adopté. Notamment la version modifiée du l'algorithme CAS (contexte-aware stemming), lui-même basé sur porter Stemmer proposés dans [12] et dont le but était de maximiser la proportion de racines (stems) significatives et aussi, l'efficacité de la recherche d'information.

L'algorithme SECAS (semantically enriched context-awere stemming) combine des fonctionnalités de l'algorithme de stemming (CAS) et le dictionnaire (wordnet) ce qui en fait une technique d'indexation conceptuelle permettant la formation racines très comparables au lemme (lemmatisation). Bien qu'il y ait une différence entre Stemming et Lemmatisation, dans le stemming, un ensemble des règles est appliqué pour former les formes de base finales sans tenir compte du contexte textuel, alors qu'en lemmatisation la compréhension du contexte des mots dans une phrases est très important avant la réduction des formes de mots peut être effectué, la base des deux méthodes est de réduire un mot variante a sa stem dans le cas de stemming et lemma dans le cas de lemmatisation.

Dans ce chapitre, nous parlons de les déférences entre CAS ,SECAS et PORTER ,et une explication détaillée de la façon dont cela fonctionne l'algorithme SECAS .Notre objectif principal est d'améliorer le rappel et la précision dans les SRI.

Malgré le fait que l'algorithme CAS (contexte-aware stemming), fournit des racines significatives et les stemmers basés sur des règles profitent de certains phénomènes de langage qui peut facilement être exprimé par des règles simples, les deux sont des tâches fastidieuses.

SECAS(semantically enriched context-awere stemming) peut être utilisé efficacement dans les étapes de prétraitement du système d'indexation des textes et de classification dans le contexte de RI et son objectif principal est de fournir des racines significatives afin d'améliorer le rappel sans diminuer la précision. SECAS est divisée en deux partie principales : la première partie est une phase de prétraitement ; où les mots vides, pluriels, et les caractères spéciaux sont supprimés. La deuxième consiste en une version améliorée de l'algorithme CAS pour obtenir un bon stemmer

2. Problématique

Les progrès des technologies de la communication et de la recherche ouvrent de nouveaux horizons et permettent la diffusion immédiate d'une grande quantité d'information.

L'énorme explosion de l'information augmente la nécessite de découvrir de nouveaux outils de recherche d'information (RI).

On a plusieurs méthodes pour réponde aux besoins de l'utilisateur (cas, porter ,secas)

Alors quelle est la différence entre ces algorithme

3. Objectifs

Le but de notre travail est améliorer les résultats et diminuer le temps d'exécution pour les algorithme porter et cas en ajoutant l'étape prétraitement ,et comparer les 3 algorithme cas ,porter et SACAS d'où le temps d'exécution et les résultats obtenus.

.

4. Organisation du mémoire

Pour atteindre les objectifs susmentionnés, nous avons organisé notre mémoire comme suit :

- Un premier chapitre abordera la notion de « Recherche d'Information plein texte » où les concepts de base de la RI seront présentés ainsi que les modèles et domaines d'application en passant par les Systèmes de Recherche d'Information.
- Un deuxième chapitre traitera la notion de « Sensibilité » au contexte », nous y définirons le contexte et ses principales composantes ainsi que son importance.
- Un troisième chapitre englobera la conception de notre solution et ses différents composants, modules d'indexation et d'appariement requête-document.
- Un quatrième et dernier chapitre parlera de nos tests et implémentation. Nous y présenterons notamment le dataset et l'environnement de développement utilisés.

Système de Recherche d'Information

Plein-texte

Chapitre 1 : Système de Recherche d'Information Plein-texte

1.1 Introduction

La Recherche d'Information (RI) est un sous domaine de l'informatique, qui concerne principalement la recherche et la récupération d'informations prédites à partir d'une base de documents volumineuse. Le défi est de pouvoir trouver les informations qui correspondent aux besoins de l'utilisateur.

Le processus de la Recherche d'Information (RI) est réalisé par un ensemble de logiciels, appelés Systèmes de Recherche d'Information (SRI). Ces systèmes ont pour objectif d'aider un utilisateur à trouver, à retrouver ou à découvrir des documents susceptibles de l'intéresser. L'utilisateur exprime son intérêt ou sa demande d'information à travers une requêtetextuelleet le SRI doit ensuite être capable de fournir à l'utilisateur des documents pertinents pour sa requête.

Plus récemment, avec l'arrivée de Web 2.0, également appelé Web social, à placer la RI face à de nouveaux défis d'accès à l'information. Il s'agit cette fois de retrouver une information pertinente dans un espace diversifié et de taille considérable. Ces difficultés ont donné naissance à une nouvelle discipline appelée Recherche d'Information sur le Web.

1.2 La Recherche d'Information

La Recherche d'Information (RI) fut fondée peu de temps après l'avènement des premiers ordinateurs, et constitue certainement la plus ancienne application de l'informatique à l'accès aux documents électroniques [4].

Le terme RI a été utilisé pour la première fois officiellement lors du lancement de la conférence RIAO (Recherche d'Information Assistée par Ordinateur) en 1985, à Grenoble. On parlait auparavant de « recherche documentaire »

Ou « d'informatique documentaire » [4].

1.2.1 Définitions

Plusieurs définitions de la recherche d'information ont vu le jour ces dernières années, nous citons dans ce contexte les trois définitions suivantes :

- ***Définition 1****:* La recherche d'information est une activité dont la finalité est de localiser et de délivrer des granules documentaires à un utilisateur en fonction de son besoin en informations [1].
- ***Définition 2*** : La recherche d'information est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche et la sélection d'information [2].
- ***Définition 3*** *:* La recherche d'information est une discipline de recherche qui intègre des modèles et des techniques dont le but est de faciliter l'accès à l'information pertinente pour un utilisateur ayant un besoin en information [3].

Toutes ces définitions se retrouvent dans un même contexte où la RI a pour objet d'extraire d'un document ou d'un ensemble de documents les informations pertinentes qui reflètent un besoin d'information

1.2.2 Concepts de base de la recherche d'information

Le but principal de la RI est de réalisé un processus qui permet de retrouver les documents pertinents à une requête d'utilisateur, à partir d'une base volumineuse de documents [5].

Dans cette définition, il y a trois notions clés [4].: documents, requête, pertinence.

- **Document :** Un document peut être un texte, un morceau de texte, une page Web, une image, une bande vidéo, etc. On appelle document toute unité qui peut constituer une réponse à une requête d'utilisateur
- **Requête :** Une requête exprime le besoin d'information d'un utilisateur. Elle est en général de la forme d'une phrase qui contient des mots-clés qui présentent aux besoins de l'utilisateur
- **Pertinence :** Le but de la RI est de trouver seulement les documents pertinents. La notion de pertinence est très complexe, mais de façon générale, dans un document pertinent, l'utilisateur doit pouvoir trouver les informations dont il a besoin.

Par ailleurs, La RI est le processus par lequel une collection de données est représentée, stockée et recherchée dans le but de trouver une réponse à une demande d'utilisateur [6].

Ce processus implique diverses étapes, à commencer par la représentation des données et se terminant par le retour des informations pertinentes à l'utilisateur. L'étape intermédiaire comprend les opérations d'indexation, de filtrage, de recherche, de mise en correspondance et de classement [8].

Pour réaliser cette étape il faut d'abord réaliser les sous-processus suivants " [6] " :

- **Indexation :** c'est la présentation des documents sous forme de contenu résumé.
- **Filtrage :** Ceci est pour supprimer tous les mots vides et les mots communs.
- **Recherche** : c'est le processus de base de la RI, Il existe différentes techniques pour récupérer des documents qui répondent aux besoins des utilisateurs.

Il y a aussi trois composantes de base que le processus de recherche d'information doit prendre en charge :

-La représentation du contenu des documents

-la représentation du besoin d'un utilisateur

-la comparaison des deux représentations.

La figure 1 récapitule ces composantes.

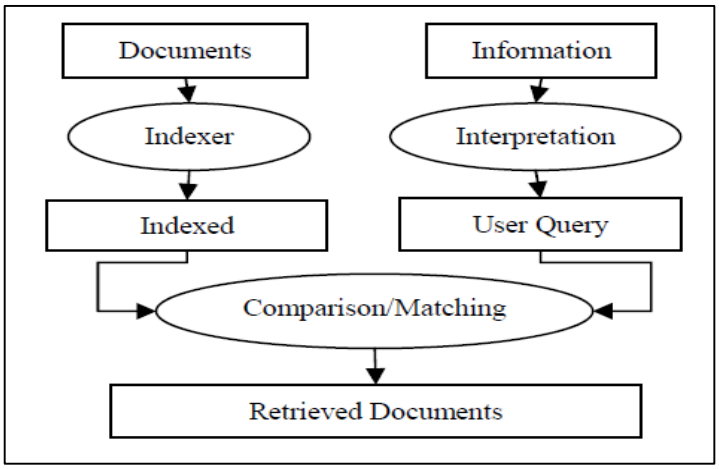


Figure 1: Processus générale de la RI

Les cases carrées représentent les données et les boîtes arrondies représentent l'enchainement du processus.

La représentation des documents est généralement appelée processus d'indexation [8], Le processus se déroule hors ligne, c'est-à-dire que l'utilisateur final du système de récupération des informations n'est pas directement impliqué, le processus d'indexation donne une représentation du document

Les utilisateurs ne recherchent pas que pour le plaisir, ils ont besoin d'informations, le processus de représentation de leurs besoins en information est souvent évoqué en tant que processus de formulation de requête [8].

La comparaison des deux représentations est appelée processus de comparaison (en anglais matching-procès) [8], les documents pertinents sont les résultats de ce processus.

L'évaluation : c'est une étape très importante dans la RI, grâce à le quelle on peut connaitre la qualité de la recherche d'information, il existe deux mesures de base pour évaluer [4] :

- **Précision :** Dans le contexte de la recherche d'information, la précision d'un algorithme de recherche décrit le rapport entre les documents pertinent et non pertinent dans un ensemble de résultats. Donc, la précision est la fraction de documents pertinents retournés (récupérés) pour une requête.
  Elle est calculée grâce à la formule :

$$\Pr ecision = \frac{\left|\{relevant\_docs\} \cap \{retrieved\_docs\}\right|}{\left|\{retrieved\_docs\}\right|}$$

- **Rappel :** Le rappel est la fraction des documents pertinents qui sont récupérés. Donc dans le contexte de la recherche d'information, plus le rappel est élevé, plus le ratio des documents pertinents dans l'ensemble de résultats par rapport aux documents du corpus est élevé.
  Il est formalisé comme suit :

$$\text{Re}call = \frac{\left|\{relevant\_docs\} \cap \{retrieved\_docs\}\right|}{\left|\{relevant\_docs\}\right|}$$

1.2.3 Les modèles de recherche d'information

On convient que les prémisses fondamentales qui forment la base d'un algorithme de classement déterminent le modèle de RI. Tout au long de cette section, nous aborderons ces différentes

notions. Cependant, avant de le faire, nous devrions indiquer clairement ce qu'est exactement un modèle de RI.

Un modèle de RI peut être considéré comme un quadruple [D, R, F, Sim (ri, dj)], où [4]:

- **D :** est un ensemble composé de vues logiques (ou de représentations) de tous les documents de la collection;
- **R :** est un ensemble composé de vues logiques (ou de représentations) des besoins des utilisateurs en Information ;
- **F :** est un Framework qui modélise les représentations des documents, des requêtes, et des relations entre eux;
- **Sim (ri, dj) :** est une fonction de similarité qui associe un nombre réel entre une requête Ri et un document Dj. Une telle similarité définie le rang d'un document par rapport au classement de tous les documents d'une collection vis-à-vis d'une requête R_i.

La figure 2 montre divers modèles RI, qui s basent de manière générale sur les composantes D, Q, F et Sim (qi, dj) avec des formalismes différents

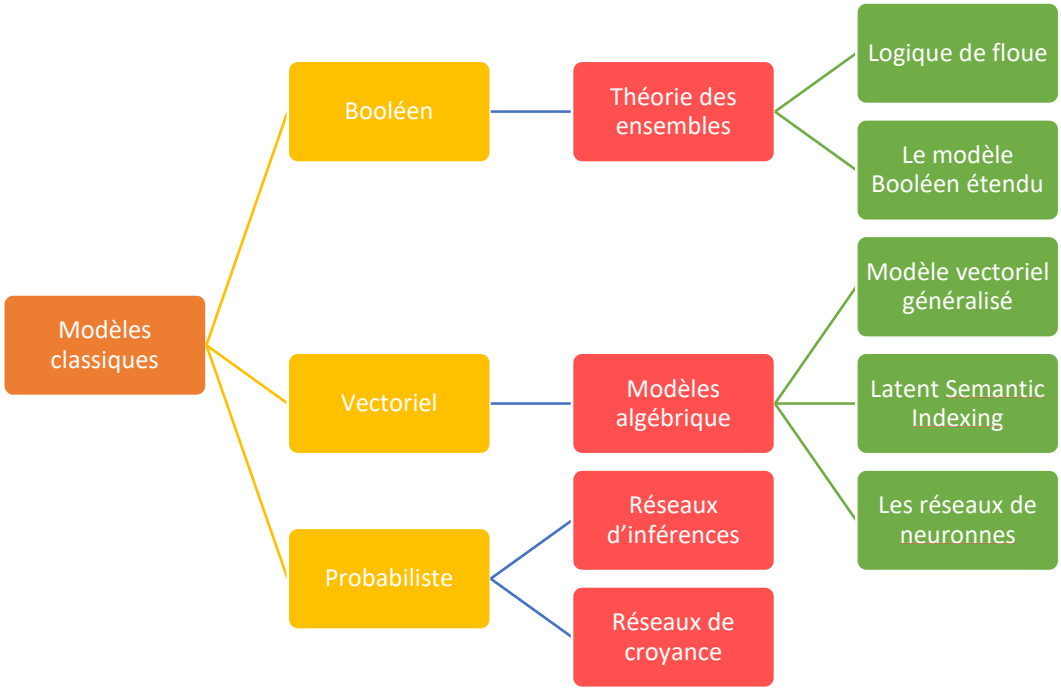


Figure 2 : Classification des modèles de RI [4].

1.2.3.1 Modèle booléen

Le modèle booléen utilise la théorie des ensembles, c'est-à-dire l'algèbre booléenne et ses trois composants AND, OR et NOT, pour la formulation de la requête, mais présente un inconvénient majeur: un système booléen ne parvient pas à classer la liste de résultats des documents pertinents [6].Dans le modèle booléen, tous les documents sont associés à un ensemble de mots ou de mots-clés distincts. Les requêtes des utilisateurs sont également représentées par des expressions de mots-clés, séparées par AND, OR, ou NOT/BUT.
La fonction de récupération du modèle booléen considère qu'un document est soit pertinent ou non pertinent.

Sur la Figure 3, les ensembles récupérés sont visualisés par les zones ombragées.

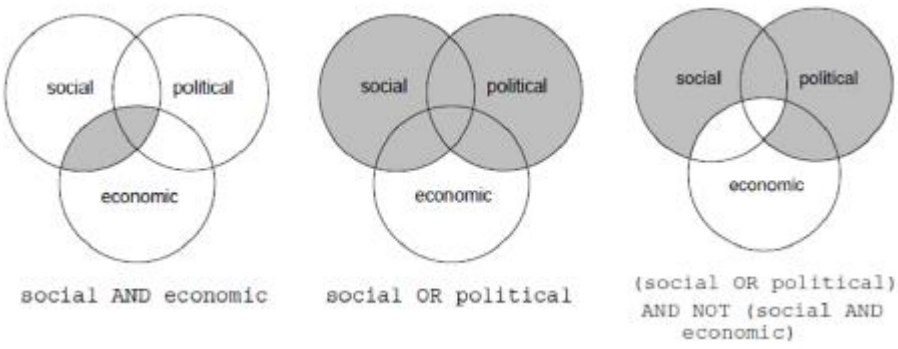


Figure 3 : Combinaisons booléennes d'ensemblesvisualisées sous la forme de diagramme de Venn.

1.2.3.2 Modèle vectoriel

Gérard Salton et ses collègues ont suggéré un modèle basé sur le critère de similarité de Luhn qui a une motivation théorique plus forte (Salton et McGill 1983) [6].

Le modèle vectoriel représente à la fois les documents et les requêtes sous forme de vecteurs dans un espace multidimensionnel, dont les dimensions consistent en mots-clés [6]. Chaque vecteur représentant des documents et des requêtes peut être construit avec des poids à terme.

Une des pondérations les plus influentes développées pour le modèle vectoriel est TF-IDF [6] (TF: TermFrequency c'est le nombre de fois où le terme apparaît dans le document et IDF :

Inverse Document Frequency c'est l'inverse du nombre de documents de la collection dans laquelle le terme apparaît.).

La similarité entre un document et une requête est calculée (avec la mesure de similarité produit interne par exemple), La figure 4 illustrer cette formule :

$$Rsv(d_j, q_k) = \sum_{i=1}^{n} w_{ij} \times w_{q_{ik}}$$

Wijest le poids du terme i dans le document ***dj***.
Wqikest le poids du terme i dans la requête $\boldsymbol{q_k}$.

1.2.3.3 Modèle probabiliste

Les modèles probabilistes appliquent des concepts et des techniques provenant de domaines tels que la logique mathématique et l'intelligence artificielle. Plus encoure les modèles probabilistes se basent sur les théories de probabilité, en particulier le principe de classement de probabilité [4,6].

La fonction la plus essentielle du modèle probabiliste est de commencer à classer les documents en fonction de leur probabilité de pertinence compte tenu de la requête de l'utilisateur.

Les documents et les requêtes des utilisateurs sont représentés par des vecteurs ~ d et ~ q qui sont des vecteurs binaires, chaque élément de vecteur indiquant si un attribut ou un terme de document apparaît dans le document ou la requête, ou non.

1.2.3.4 Discussion

Les modèles mentionnés dans cette section sont les modèles de la RI les plus étudiés dans la littérature. Par ailleurs,un grand nombre d'autres modèles ont été étudiés et utilisés dans les SRI prototypes et portent le nom de « modèles alternatifs » [4]. Ils représentent des extensions des modèles classiques, comme le montre la figure 2.

Nous pouvons noter que tous ces modèles se heurtent aux aspects intrinsèquement ambigus du langage humain, en particulier la polysémie et la synonymie. Dans ce mémoire nous proposons un modèle qui tente de surmonter ces problèmes linguistiques en faisant correspondre les requêtes à des documents basés sur la sémantique plutôt que sur la syntaxe.

1.3 Système de Recherche d'Information plein-texte

Un système de recherche d'information (SRI) est un ensemble de logiciels assurant les fonctionnalités nécessaires à la recherche d'information[4]. L'objectif d'un SRI est d'aider un utilisateur à trouver, à retrouver ou à découvrir des documents susceptibles de l'intéresser. Tout au long de cette section, nous tenterons de définir les systèmes de recherche d'information plein-texte.

1.3.1 Recherche Information plein-texte

C'est un domaine multidisciplinaire impliquant la recherche d'information, extraction d'information, analyse de texte, apprentissage automatique, visualisation, technologie de base de données et recherche d'information et la fouille de données (datamining).

La recherche d'information plein-texte est également appelée datamining, qui est une technique de recherche dans un document électronique ou une base de données textuelles, qui consiste pour le SRI à examiner tous les mots de chaque document enregistré et à essayer de les faire correspondre à ceux fournis par l'utilisateur [4].

L'ensemble des données peut être un texte sans aucune structuration (on l'appelle aussi plein-texte), il peut aussi être un texte avec une partie structurée, généralement stockée et accessible via les SRI, ou complètement structuré, qui est stockée et accessible via des systèmes de gestion de bases de données relationnelles. Chaque système de traitement possède ses propres fichiers de stockage de données et méthodes d'accès [5,4].

Aujourd'hui, la distinction entre données structurées et semi-structurées disparaît rapidement. En fait, nous ne nous intéressons plus uniquement aux données structurées et semi-structurées, mais également aux images, aux fichiers audios et aux vidéos dans le même référentiel de stockage.

Les principaux problèmes des systèmes de recherche d'informations plein-texte sont [4] :

- Ils peuvent récupérer des documents non pertinents qui incluent des termes ambigus (par exemple le mot "Apple" a deux sens fruit et entreprise).
- Ils ne peuvent pas récupérer les documents pertinents s'ils incluent des termes synonymes (par exemple tuerie et massacre deux mots avec le même sens).

1.3.2 Techniques de recherche

Il y a plusieurs techniques ou algorithmes de recherche, comprenant recherche linéaire, recherche binaire et recherche par force brute.Ces algorithmes de recherche sont décrits comme suit[6,4] :

1.3.2.1 Recherche linéaire

Il s'agit d'un algorithme simple à utiliser qui recherche un mot spécifique ou un mot clé dans une liste ou un tableau et vérifie tous les éléments de la liste à la fois.
La recherche linéaire est une des techniques les plus simples. L'un des inconvénients les plus importants est la lenteur de traitement des listes ordonnées. Ce type d'algorithmes est également appelé recherche séquentielle.

1.3.2.2 Recherche binaire

Il s'agit de trouver la position d'une valeur d'entrée particulière, à savoir «la clé de recherche » dans un tableau trié selon une valeur de clé. A chaque étape, l'algorithme compare la valeur de la clé de recherche avec la valeur de clé de l'élément central du tableau. Si les clés correspondent, un élément correspondant a été trouvé et son index, ou sa position, est retourné.

Sinon, si la clé de recherche est inférieure à la clé de l'élément central, l'algorithme répète son action sur le sous-tableau situé à gauche de l'élément central ou, si la clé de recherche est plus grande, sur le sous-tableau de droite.

Si le tableau restant à rechercher est vide, la clé est introuvable dans le tableau et une indication spéciale "non trouvé" est retournée, appelle recherche de dichotomie.

1.3.2.3 Recherche par force brute

C'est un algorithme qui consiste principalement à essayer toutes les solutions possibles. Par exemple, pour trouver le maximum d'un certain ensemble de valeurs, on consulte toutes les valeurs. Cet algorithme utilise une technique très simple à appliquer qui consiste à donner des solutions si elles existent [6].

1.4 Classement dans la recherche d'information

Le classement des résultats des requêtes est l'un des problèmes fondamentaux de la RI, en partie à cause du rôle central du processus de classement dans les moteurs de recherche [4].

Une grande attention a été portée à la recherche et au développement de technologies de classement. Ou cette dernière pose des problèmes dans d'autres domaines de la RI, comme le filtrage collaboratif.

Parfois, on a besoin de classer les documents uniquement en fonction de leur pertinence par rapport à la requête. Dans d'autres cas, nous devons prendre en compte les relations de similarité et de diversité entre les documents dans le processus de classement.

Pour résoudre le problème de la récupération de documents, plusieurs modèles de classification expérimentaux ont été proposés et utilisés dans la recherche documentaire.Récemment, il est devenu possible de tirer parti des technologies d'apprentissage automatique pour créer des modèles de classement efficaces.

Plus précisément, nous appelons les méthodes qui apprennent à combiner des fonctionnalités prédéfinies pour le classement au moyen de méthodes d'apprentissage discriminant « Learning-to-Rank » [4].

Une architecture d'un moteur de recherche utilisant la technique du learning tout rank est illustrée dans la figure 4.

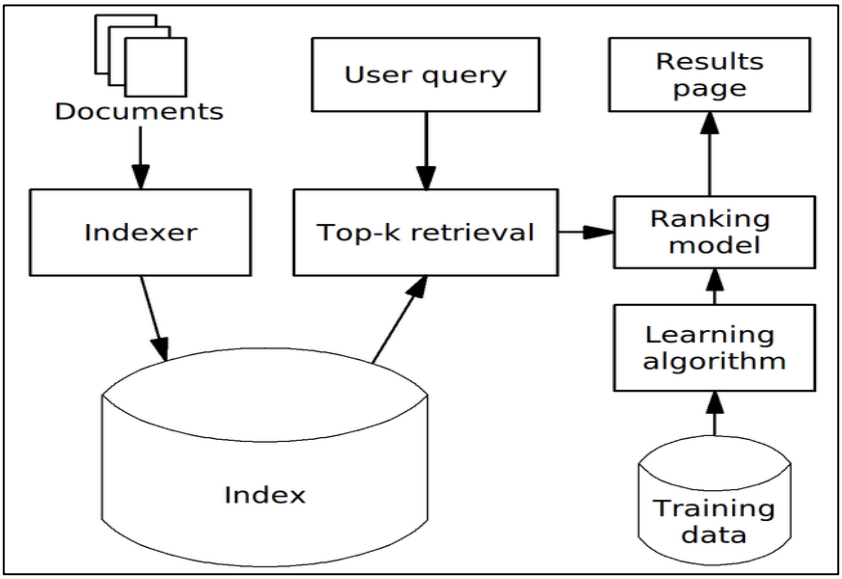


Figure 4: Exemple de moteur de recherche avec Learning-to-Rank[4].

1.5 Traitement automatique de la langue et la RI

Plusieurs techniques de traitement automatique de la langue (TAL), notamment Tokenisation, Stemming, Lemmatisation, normalisation, Part-of-speech tagging, entre autres, ont été utilisés dans la recherche d'information.

Le processus principal de RI abordé dans ce mémoire est la récupération de documents pertinents.Plusieurs autres tâches de RI utilisent des techniques très similaires, par exemple la collecte de documents, le filtrage, la détection de nouveaux événements et la détection de liens. Ces taches peuvent faire appel à certain fonction TAL de manière similaire à la RI.

Le traitement de texte comporte deux techniques principales: la Tokenisation et la Normalisation.Deux autres étapes ont démontré leur efficacité pour améliorer la pertinence des résultats retournés par le SRI : suppression des mots-vide et Stemming ou Lemmatisation. Presque toutes les applications RI suppriment les mots vides avant de traiter des documents et les requêtes.

1.5.1 Tokenisation

C'est le processus consistant à découper les espaces blancs et à supprimer les caractères de ponctuation. Chaque terme résultant de cette tâche est appelé token [4].

Exemple:

Input : *"* Bonjour monsieur, est ce que le directeur est là ?. *"*

Output : |Bonjour| |monsieur| |est| |ce| |que| |le| |directeur| |est| |là|.

1.5.2 Suppression des mots-vide

Les mots vides sont les mots extrêmement communs et non significatifs figurant dans un texte. Ils sont totalement exclus du dictionnaire [4]. La stratégie de suppression est la suivante:

a) Trier les termes par fréquence.

b) Ajouter les termes les plus fréquents à la liste des mots-vide

   Exemple :

   anglais :the ,or ,a …..

   Français :le ,la, de ,je……

1.5.3 Normalisation

C’est un processus qui convertit une liste de mots en une séquence plus uniforme. Par exemple il traite la capitalisation (ChAt→ chat) et suppression des accents et des signes diacritiques [4].

Exemple : [4]. Exemple résumé →resume, naïve → naive

1.5.4 Stemming et Lemmatisation

Le stemming et la Lemmatisation peuvent être considérées comme des techniques de normalisation [4]. Le but principal de ces techniques est de réduire les formes flexionnelles et les formes parfois dérivées d’un terme à une forme de base commune.

1.5.4.1 Stemming

C’est un processus de transformation des termes en leur radical ou racined’atteindre cet objectif correctement la plupart du temps.

1.5.4.2 Lemmatisation

C’est un processus précis qui utilise un dictionnaire et une analyse morphologique des mots, visant à renvoyer la forme de base ou de dictionnaire d'un mot.

1.6 Domaine d’application des systèmes de RI

Les SRI ont été d’abord développés pour aider à gérer l'énorme quantité d’information. Plusieurs universités, entreprises et bibliothèques publics utilisent maintenant des systèmes de RI pour donner accès à des livres, des journaux et d’autres documents. La recherche d'information est aujourd'hui utilisée dans de nombreux domaines. Certaines applications de SRI sont discutées dans cette section

1.6.1 Moteur de recherche

Le moteur de recherche une des applications les plus pratiques dans les systèmes de RI, il permet de retrouver des données sur le web (documents, fichier texte, vidéo, image, ...etc.) associées à des mots quelconques. Les moteurs de recherche ne s'appliquent pas qu'à Internet : certains moteurs sont des logiciels installés sur un ordinateur personnel. Ce sont des moteurs dits desktop qui combinent la recherche parmi les fichiers stockés sur le PC et la recherche dans les sites Web [5].

1.6.2 Bibliothèque numérique

Une bibliothèque numérique est un type de bibliothèque dans laquelle les collections sont stockées dans des formats numériques et ces formats sont accessibles par des systèmes informatiques. L'information numérique peut être stockée localement ou accessible à distance via d'internet [5,6,4].

1.6.3 Recherche multimédia

Ce type de recherche peut être appliqué par des interfaces de recherche multimédia, ce qui signifie que d'autres types de média sont également inclus pour différentes informations sur la recherche de texte. Par exemple, le système de récupération d'images est un SRI multimédia installé dans l'ordinateur pour la navigation, la recherche et la récupération de certaines images parmi grande une collection d'images numériques [4,6].

1.6.4 Filtrage d'information

Le système de filtrage d'information comprend de nombreux outils permettant à l'utilisateur de récupérer les informations les plus pertinentes [6].

Les filtres d'information sont également utilisés pour gérer et structurer l'information de manière juste et intelligible.

1.6.5 Annuaire

L'annuaire (ou directory en anglais) est une liste de liens subdivisés en catégories suivant une structure en arbre, accompagnée d'une brève description. Bien que ce procédé fût pionnier en la matière, il tend à disparaître. En effet, le fait de devoir sélectionner les catégories dans lesquelles on recherche suppose que l'on sache exactement où chercher. Et on peut se demander où se positionne le site qui appartient à plusieurs catégories. Mais à cette question, les moteurs utilisant ce procédé vous répliqueront qu'ils se trouvent dans toutes celles susceptibles de correspondre. Néanmoins, on doit reconnaître aux annuaires un gros avantage, celui de mettre en quelque sorte dans le contexte, ainsi les recherches dans la base de données sont diminuées, en plus d'obtenir des résultats plus pertinents. Les annuaires sont donc des outils basés sur le recensement humain de l'information. Ils signalent des sites et des ressources de l'Internet comme un catalogue de bibliothèque signale des livres ou bien encore comme les pages jaunes signalent des entreprises. On distingue dans ce contexte deux catégories d'annuaires [5].

1.6.6 Comparaison entre les Moteurs de Recherche et les Annuaires

Les moteurs de recherche et les annuaires sont des outils utilisés par les internautes pour rechercher des informations sur le Web. La différence réside dans la manière dont ils obtiennent et organisent leurs informations.

Les moteurs de recherche utilisent des agents logiciels automatisés appelés spiders, crawlers, robots et bots. Ces robots sont les chercheurs de contenu sur Internet, ils ramènent l'information à l'index du moteur de recherche [9]. Lorsque vous utilisez Google, vous ne recherchez pas réellement sur le Web, vous recherchez l'index Web de Google. Ces outils sont des éléments clés dans le fonctionnement des moteurs de recherche. Les moteurs de recherche utilisent des algorithmes complexes pour déterminer les pages Web les plus susceptibles de répondre aux questions que vous posez et les renvoyer sur les pages de résultats des moteurs de recherche.

Par contre, dans les annuaires les propriétaires de sites soumettent leurs sites aux annuaires (parfois moyennant des frais, parfois gratuitement) et les éditeurs humains déterminent la valeur du site et déterminent si celui-ci doit être inclus dans l'annuaire. Bien que les annuaires soient devenus démodés (même l'annuaire de Yahoo est maintenant caché sur Yahoo.com sous l'onglet "plus"), il est toujours avantageux d'y figurer [5,9]. Le fait d'être répertorié dans un annuaire important améliore la visibilité d'un site web, organisation, entreprise donnée.

Enfin, la distinction est importante entre les annuaires et les moteurs de recherche. Le tableau 1 résume les avantages et les inconvénients de ces outils suivant [5] :

Tableau 1: ***Comparaison entre annuaires et moteurs de recherche*** **[5]**

Moteur de recherche	**Annuaire**
Indexation des termes par des bots ou des robots	Indexation de sites par des bibliothécaires
Recherche en texte intégral sur des pages web	Recherche sur des sites et sur des catégories
Avantages : plus d'exhaustivité et mise à jour plus rapide	Avantages : choix des informations, classement raisonné par catégories et
Inconvénients : capture de pages web sans classement raisonné	Inconvénients : moins d'exhaustivité et mise à jour moins rapide

À retenir : La recherche par mots-clés donne de meilleurs résultats sur les moteurs	À retenir : L'exploration des catégories s'avère souvent plus fructueuse que celle des sites.

I.7 Conclusion

Au début de ce chapitre, on a étudié les concepts, les techniques et les modèles classiques de la recherche d'information, dont la plupart s'adressent principalement au domaine dans son sens le plus large. Et puis nous avons abordé les concepts des Systèmes de RI plein-texte. La RI plein-texte se présente aujourd'hui sous la forme d'assistants personnels intelligents.

Les Système de RI actuels rencontrent de nombreux problèmes, comme en témoigne leur incapacité à traiter les demandes en temps réel et à présenter adéquatement les résultats. Compte tenu de ces insuffisances, il est impératif de développer un système de RI qui tiendra compte du contexte de la tâche de RI.

Effectivement, identifier le sens des mots dans leur contexte n'est pas difficile pour les interprètes humains, mais reste un défi, même pour les machines les plus avancées, puisqu'un mot peut possède plusieurs sens qui indiquent différentes significations dans différents contextes.

De plus, même si les efforts d'évaluation des RI prennent l'utilisateur en considération en incluant des jugements de pertinence prédéfinis et diverses méthodes d'évaluation, leur nature est limitée. Comme les humains sont diversifiés, les utilisateurs du système RI le sont aussi. En conséquence, l'interaction de l'utilisateur avec les systèmes RI présente des comportements divers, ce qui manque à la conception et à la mise en œuvre des systèmes pertinents. Les études classiques supposent qu'un utilisateur moyen interagit avec un système de récupération de manière prévisible et régulière. Cependant, les utilisateurs sont divers et pas toujours prévisibles [4].

Le contexte est l'une de nos préoccupations majeures et nous en discuterons en détail au chapitre2.

Sensibilité au contexte

Chapitre 2 : sensibilité au contexte

2.1 INTRODUCTION

Après l'arrivée et l'installation durable des ordinateurs dans nos foyers, l'informatique est appelée à être de plus en plus présente au quotidien. L'explosion de la téléphonie mobile, l'arrivée sur le marché des ordinateurs portables et maintenant des ultra-portables, sorte de fusion entre un ordinateur portable et un organisateur, donnent une idée de ce qui peut nous attendre en informatique dans les années à venir : de plus en plus de mobilité et de communication et des utilisateurs s'attendent à ce que leurs appareils répondent à des attentes motivées aussi par l'environnement.

Ces projets sont dans les cartons des laboratoires depuis une dizaine d'années, sous le nom d'abord d'informatique ubiquitaire puis plus spécifiquement d'informatique contextuelle. La présente bibliographie a pour intention de présenter un état de l'art du domaine en analysant certains des articles clés publiés par les équipes de chercheurs qui ont mis pied dans ces nouvelles formes d'interactions homme machine.

Dans un premier temps, nous présenterons une synthèse des définitions données pour l'informatique contextuelle. Ensuite, nous parlerons de catégories de contexte, les caractéristiques de contexte et l'importance du contexte dans la RI.

2.2 Contexte et sensibilité au contexte

Dans cette partie, nous parlerons sur le contexte en détail car il joue un rôle très important dans le SRI, qui est de donner des informations précisés et exactes à l'utilisateur.

2.2.1 Contexte

Citons les quelques définitions de contexte suivantes :

- Schilit et Theimer étaient les premiers a avoir introduit le terme "context-aware" (sensible au contexte) mais leur définition du contexte ne porte que sur des variables comme la localisation, l'identité des personnes proches et des objets ainsi que les changements dans ces objets. Avec cette définition, il est impossible pour un élément pris dans l'environnement, de savoir avec certitude s'il fait partie ou non du contexte.
- Le contexte est l'ensemble de toutes les informations qui peuvent être utilisées pour caractériser la situation d'une entité. Une entité pouvant être un lieu, un objet de l'environnement considère comme utile à l'interaction entre l'utilisateur et le système [11].

➢ Ils définissent ainsi le contexte d'interaction comme étant l'intersection entre les connaissances contextuelles de l'ordinateur et celles de l'utilisateur ou l'idée que l'utilisateur se fait des connaissances contextuelles du système (figure5) [11].

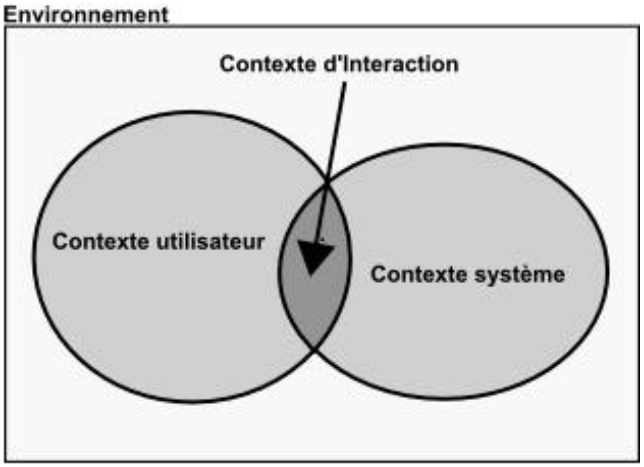


Figure 5: **Définition du contexte d'interaction** [11].

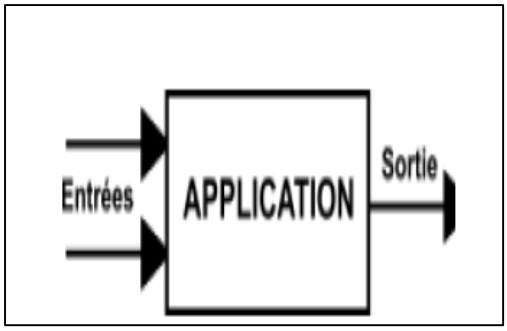
➢ Le contexte fait références aux circonstances dans lesquelles l'événement se produit, ce qui inclut tous les facteurs internes et externes associés à un processus de recherche particulier ceci afin de transmettre des informations précises à l'utilisateur [4].

2.2.2 Catégorie de contexte et caractéristiques

Bien qu'il existe plusieurs définitions de contexte, elles partagent les mêmes catégories et caractéristiques. Dans cette section nous aborderons une variété de catégorie de contexte et de caractéristiques utilisées pour différentes applications.

Selon (schilit and theimer 1994), le contexte est caractérisé comme suit [10] :

- **Localisation** : c'est la position de l'entité ;
- **L'identité de l'utilisateur** : chaque entité a un identifiant unique ;
- **Activité** : les propriétés de l'entité ;
- **Temps** : utilise Our définir la situation avec précision.



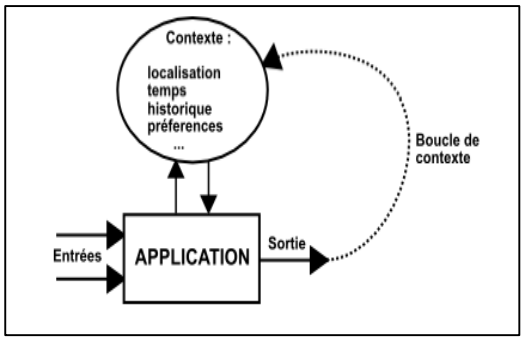


Figure 6 : Application sans contexte [11] **Figure 7**: **Application avec contexte [14]**

Les notions de temps, d'identité, d'activité et de localisation sont les principaux types de contexte permettant de caractériser la situation d'une entité particulière, en fait toute variable contextuelle est utile pour éclaircir une ou plusieurs des questions "**Quand** ?" "**Où** ?" "**Quoi** ?" "**Qui**" (c'est à dire, ce que l'utilisateur fait).

Le système va utiliser ces informations catégorisées pour répondre au "**Pourquoi**" de l'occurrence d'une situation. [13]

Un utilisateur équipé d'un système contextuel de type guide touristique, par exemple lors d'une visite à la ville de TIPAZA, va s'approcher de la mer et donc il va voir apparaitre sur ce guide des informations sur la mer. Le concepteur du système répond à la question 'pourquoi l'utilisateur s'approche-t-il?' en lui envoyant des information sur la mer .

La variable contextuelle utilisée est la localisation de l'utilisateur (ville de Tipaza) mais aussi son environnement, c'est à dire que la mer fait aussi partie du contexte puisqu'il influe sur la réponse du système, donc l'environnement répond à une question fondamentale de ce qui se passe dans la situation [15].

Dans cette catégorisions initiale, nous avons un système simple à deux niveaux : **contexte primaire** et **contexte secondaire** [15], les quatre éléments primaires du contexte (**localisation, l'identité, temps, activité**) se situent au premier niveau, tous les autres types de contexte sont au deuxième niveau,

Le contexte primaire est toute information obtenue de manière explicite (saisi utilisateur, ou capteur) et récupérée sans utiliser le contexte existant.

Le contexte secondaire est toute information qui est dérivée par la manipulation du contexte primaire. Les élément de contexte secondaires partagent une caractéristique commune : ils peuvent être indexés par un contexte primaire car ils sont des attributs de l'entité avec le contexte

primaire [15], par exemple le numéro de téléphone d'un utilisateur est un élément de contexte secondaire et il peut être obtenu en utilisant l'identité de l'utilisateur en tant qu'index dans un espace d'information tel qu'un annuaire téléphonique.

Cette caractérisation aide les concepteurs à choisir le contexte à utiliser dans leurs application [15].

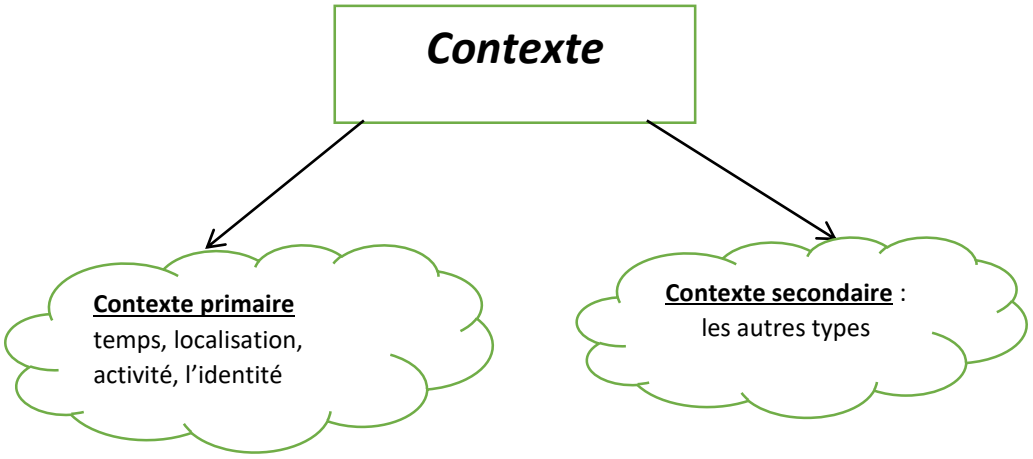


Figure 8 : définition de types de contexte

Il existe de nombreuses façons de classer le contexte en différentes catégories. Ci-dessous les catégories de contexte proposées par [16] :

- C**ontexte informatique** : comprend la connectivité réseau, la communication, la bande passante et les ressources informatiques locales telles que les imprimantes, les écrans...etc.
- C**ontexte de l'utilisateur** : peut-être le profil de l'utilisateur, son emplacement, sa situation sociale.
- C**ontexte physique** : peut-être les conditions de circulation, la température…etc.
- C**ontexte temporel** : comprend l'heure du jour, la semaine, le mois et/ou la saison de l'année.
- C**ontexte historique** : est le stockage du contexte existant à différents moments.

Un autre moyen de classer les types de contexte est de le faire selon contexte « *externe* » et « *interne* » [17][18][19] ou encore contexte « *physique* » et « *logique* » respectivement [20]. Le contexte physique (externe) fait référence au contexte qui peut être mesuré par le matériel

capteurs tels que la position, la lumière, le son…etc. D'autre part le contexte logique (interne) est spécifié par l'utilisateur, par exemple son objectif…etc.

De nos jours, le contexte social est devenu populaire parce qu'il y a eu les demandes croissantes d'application socialement conscientes. Le contexte social est utilisé pour être défini comme la personne à proximité ou le groupe auquel l'utilisateur appartient.

Le contexte social est un ensemble d'informations provenant de sources directes ou indirectes. Le contexte social joue un rôle important dans la sécurité et la santé publique comme dans la détection automatique des foules, analyse criminelle, infection par la maladie...etc [21] .

Maintenant que nous avons défini le contexte, nous pouvons commencer réfléchir à comment utiliser le contexte.

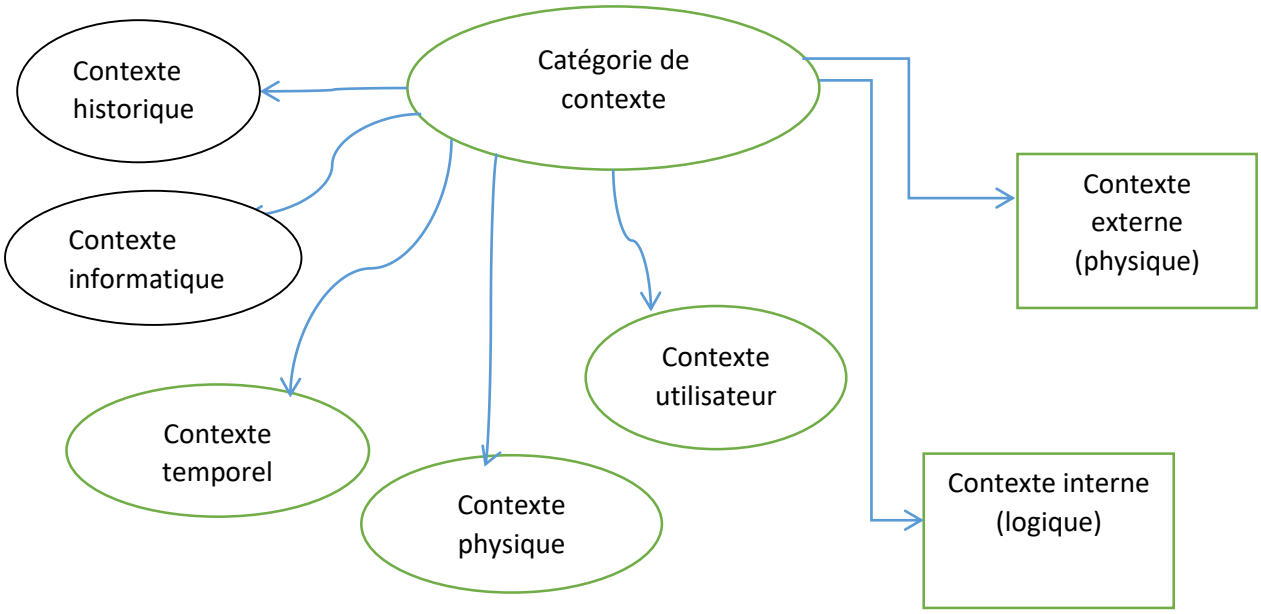


Figure 9 : **représentation des catégories de contexte**

Les carrés représentent les catégories de contexte externe et interne

Les cercles représentent les catégories de contexte historique ,informatique,temporel,physique et utilisateure

2.2.3 Sensibilité au contexte

La sensibilité au contexte, c'est la possibilité pour un système ou un composant de système de recueillir à tout moment des informations sur son environnement et d'adapter ses comportements en conséquence. L'informatique sensible au contexte ou contextuelle s'appuie sur les logiciels et le matériel pour collecter et analyser automatiquement les données facilitant les réponses.

Salbert et al. [22] définissent la sensibilité au contexte comme étant la meilleure capacité d'un système à agir en temps réel avec des données provenant du contexte. Dey [11] limite sa définition à l'interface homme machine sans prendre en compte l'application en elle-même.

D'autres définitions sont plus orientées vers l'adaptation au contexte : Brown [23] dit d'une application sensible au contexte qu'elle doit automatiquement extraire de l'information ou effectuer des actions en fonctions du contexte utilisateur détecté par les capteurs.

2.2.5 Exemples d'utilisation du contexte

Sensay (suggéré par Siewiorek, Smailagic, Furukawa, Moraveji, Reiger et Shaffer avec Sensay) est un téléphone mobile qui modifie son comportement en tenant compte de l'état utilisateur et environnement. S'adapte dynamiquement au changement de contexte et est également connue notifier l'appelant (appelant) de l'état de l'utilisateur pour spécifier le contexte [28].

2.3 L’importance de la sensibilité au contexte dans RI

La principale raison derrière l’introduction de la sensibilité au contexte est la simplification de l’expérience utilisateur. Ses avantages sont les suivants :

- Diminuer le nombre de commandes que l’utilisateur doit connaître pour un certain niveau de productivité.
- Réduire le nombre de clics ou de frappes nécessaires pour effectuer une opération particulière.
- Réduire le nombre d’options requises à l’écran en même temps.

D’autres raisons peuvent être résumés ainsi :

- **Tâche de recherche** : Les personnes ont besoin d’aide pour leurs activités.

Ainsi, le contexte peut être utilisé pour : Personnaliser les services et les informations pour l'utilisateur ou encore pour recommander des ressources, livres, vidéos, articles, blogs...etc.

- **Aspects environnementaux** : le besoin de l'utilisateur peut ne pas être suffisamment spécifique et peut être modifié plusieurs fois au cours du processus de recherche. Ainsi, L'ambiguïté et la volatilité conduisent à la nécessité de s'adapter surtout dans les conditions d'interaction entre l'utilisateur et les systèmes.
- **Technologie** : une grande quantité de données a conduit à l'émergence nouvelles applications basée sur la connaissance des préférences des utilisateurs, l'informatique contextuelle et les services de réseautage social. De même, des améliorations de haute technologie mise à jour : Contacts tactiles, Connexions 3G (4G, 5G ...), GPS ...,En particulier, l'utilisation généralisée des téléphones mobiles, l'émergence de livres, de tablettes et de téléphones intelligents dans un nouveau monde où l'utilisateur peut interagir avec plus de personnes à Plus de sites.[4]

2.5 Conclusion

Après avoir donné des éléments de définitions au contexte et à l'informatique sensible au contexte, catégories de contexte, caractéristiques de contexte, ainsi que l'importance de la sensibilité au contexte dans RI.

Dans le chapitre suivant, nous présentons la conception de notre contribution à un système de recherche d'information sensible au contexte.

Présentation des algorithme de test

Chapitre 3 : présentation des algorithme des test

3.1 Introduction

Dans ce chapitre, nous parlons des algorithmes que nous avons adopté. Notamment la version modifiée du l'algorithme CAS (contexte-aware stemming), lui-même basé sur porter Stemmer proposés dans [12] et dont le but était de maximiser la proportion de racines (stems) significatives et aussi, l'efficacité de la recherche d'information.

L'algorithme SECAS (semantically enriched context-awere stemming) combine des fonctionnalités de l'algorithme de stemming (CAS) et le dictionnaire (wordnet) ce qui en fait une technique d'indexation conceptuelle permettant la formation racines très comparables au lemme (lemmatisation). Bien qu'il y ait une différence entre Stemming et Lemmatisation, dans le stemming, un ensemble des règles est appliqué pour former les formes de base finales sans tenir compte du contexte textuel, alors qu'en lemmatisation la compréhension du contexte des mots dans une phrases est très important avant la réduction des formes de mots peut être effectué, la base des deux méthodes est de réduire un mot variante a sa stem dans le cas de stemming et lemma dans le cas de lemmatisation.

Dans ce chapitre, nous parlons de notre proposition d'algorithme stemming efficace qui intègre également les avantage des algorithmes de lemmatisation. Notre objectif principal est d'améliorer le rappel et la précision dans les SRI.

Malgré le fait que l'algorithme CAS (contexte-aware stemming), fournit des racines significatives et les stemmers basés sur des règles profitent de certains phénomènes de langage qui peut facilement être exprimé par des règles simples, les deux sont des tâches fastidieuses. A cet égard , nous avons développé un algorithmes qui a les avantages d'un stemmer qui utilise la syntaxe ainsi que la connaissance sémantique pour réduire les erreurs d'indexation. SECAS(semantically enriched context-awere stemming) peut être utilisé efficacement dans les étapes de prétraitement du système d'indexation des textes et de classification dans le contexte de RI et son objectif principal est de fournir des racines significatives afin d'améliorer le rappel sans diminuer la précision. SECAS est divisée en deux partie principales : la première partie est une phase de prétraitement ; où les mots vides, pluriels, et les caractères spéciaux sont

supprimés. La deuxième consiste en une version améliorée de l'algorithme CAS pour obtenir un bon stemmer.

3.2 Travaux connexes

Le stemming est une phase de prétraitement importante dans la plupart des applications de Textmining et de Traitement du Langage Naturel.

De plus, il a été prouvé que l'utilisation de l'option stemming dans un IRS peut améliorer de nombreuses autres tâches connexes, telles que la traduction automatique et analyse des sentiments. La création de racines est utilisée pour permettre la mise en correspondance de requêtes et de documents systèmes de recherche d'informations basés sur des mots clés. ". À cet égard, l'objet de la radicalisation est de réduire les différentes variantes grammaticales d'un mot (nom, adverbe, adjectif, verbe, etc.) dans une forme de base commune (racine) appelée «stem» (racine en Français); en supposant que les mots qui partagent la même racine, possèdent le même sens.

Cette section présente la définition des algorithmes de stemming et de leur problèmes.

3.2.1 Le concept de stemming

Dans les SRI traditionnels, une première approche pour donner des résultats en réponse à une requête consistait à aller chercher dans tous les documents du corpus mot par mot et puis, on classait les documents en fonction du nombre de leurs mots communs avec la requête d'entrée. Cette approche prenait beaucoup de temps et donnait des résultats inexacts. Donc pour surmonter ces inconvénients et augmenter la précision et la pertinence des résultats de recherche, les techniques du stemming se sont avérées satisfaisantes. L'idée du Stemming est de réduire les mots à leur racine linguistique en suppriment les préfixes et les suffixes [37-35].

Par exemple, un chercheur qui saisit le terme Happiness dans sa requête, il est probable qu'il soit également intéressé par des variantes du même mot telles que Happy.

La figure 4.1. montre différentes variantes syntaxiques pour le mot « Program ».

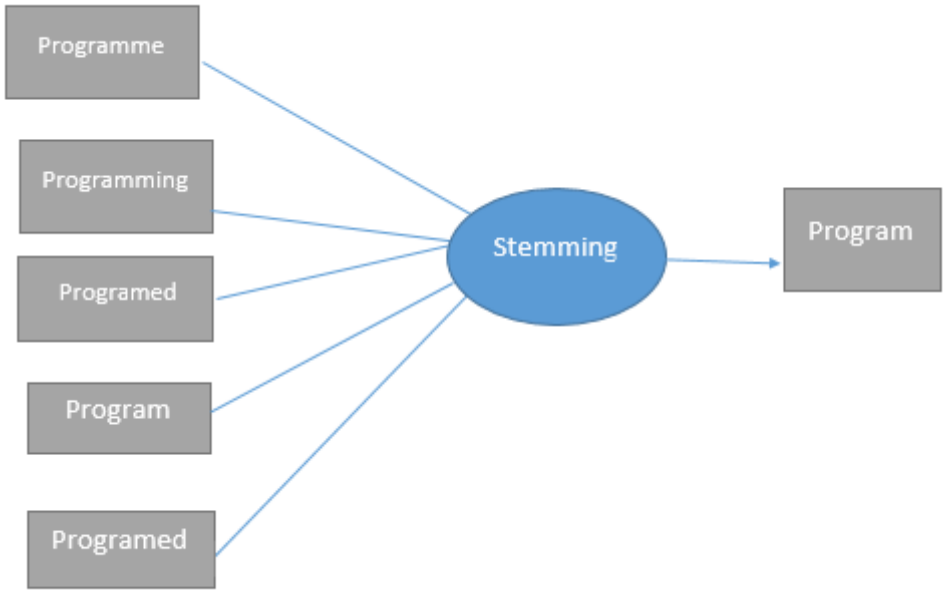


Figure 10 : Exemple illustrant le principe de stemming.

Le stemming permet d'avoir le stem d'un mot avec deux méthodes :

1. Des stemmers à base de règles (suppression des affixes). Ces algorithmes sont rapides, utilisent peu de mémoire et donnent de bons résultats avec les mots réguliers. Mais ils ne fonctionnent pas très bien avec les mots irréguliers.
2. Des stemmers à base de dictionnaire: ces algorithmes se basent sur la recherche des racines dans une ressource externe appelée dictionnaire (comme la base lexicale Wordnet par exemple). Les algorithmes à base de dictionnaire sont capables de résoudre les problèmes de synonymie présents dans n'importe quelle langue.

3.2.2 Problème de stemming

La racine d'un mot est définie comme suit « la racine d'un mot est sa forme la plus élémentaire qui peut ou non avoir une interprétation sémantique» [33]. Malheureusement, aucun algorithme de stemming n'est parfait parce que dans les documents Anglais par exemple, les informations sur les termes originaux pourraient être perdus.

Gormley and tong [36] résument les problèmes de stemming comme suit :

- Il arrive que deux mots ayant le même sens ne puissent être réduits à la même racine.
- Il arrive que deux mots de significations distinctes mènent à confusion dans la mesure où ils auront les mêmes racines. Par exemple : « general » et « generate » possèdent tous les deux la racine « gener ».

3.3 Algorithme utilise

une solution hybride synergique comme algorithme de stemming combine des fonctionnalités de stemmers algorithmiques et de stemmers dictionnaires. En effet, notre principal objectif est de maximiser la proportion de stems significatifs (racines de mots), sans compromettre les autres critères de performance. En outre, le même mot peut avoir deux significations. Pour surmonter ces cas, nous avons proposé un algorithme qui combine les avantages des dictionnaires et ceux des stemmers. En fait, l'algorithme général propose des mots racines comparable à des lemmes. Bien que la lemmatisation et la racinisation visent à normaliser les mots liés en identifiant leurs représentations canoniques, le processus de lemmatisation est plus compliqué dans la mesure où il faut comprendre le contexte dans lequel un mot est utilisé pour prendre une décision à propos de sa signification. En effet, la lemmatisation essaierait de distinguer les différents sens des mots, tout en évitant de les confondre incorrectement. L'algorithme proposé est une version améliorée de l'lagorithme CAS dédié à la langue anglaise. Malgré le fait que dans ce dernier, les racines dérivées sont linguistiquement correctes par rapport à bon nombre de racines obtenues grâce à l'algorithme de Porter), nous pouvons remarquer qu'il est parfois impossible de confondre les mots avec leur vraie racine. En outre, il ne détecte pas la bonne racine dans le cas de verbes irréguliers ou de certains noms pluriels irréguliers aussi.

Dans cette sous-section, nous décrivons notre algorithme d'indexation basé sur le contexte et sémantiquement enrichi (SECAS, de l'Anglais Semantically Enriched Context-Aware Stemming Algorithm)[1].

3.3.1 la différence entre les algorithmes

Porter : consiste à supprimer les suffixes et les préfixe indépendamment du sens du mot ou de sa place dans la phrase

Voila les règles de porter

```
Step 1a

    SSES -> SS                          caresses  ->  caress
    IES  -> I                           ponies    ->  poni
                                        ties      ->  ti
    SS   -> SS                          caress    ->  caress
    S    ->                             cats      ->  cat

Step 1b

    (m>0) EED -> EE                     feed      ->  feed
                                        agreed    ->  agree
    (*v*) ED  ->                        plastered ->  plaster
                                        bled      ->  bled
    (*v*) ING ->                        motoring  ->  motor
                                        sing      ->  sing

Step 1c

    (*v*) Y -> I                    happy         ->  happi
                                    sky           ->  sky

Step 2

    (m>0) ATIONAL ->  ATE           relational     ->  relate
    (m>0) TIONAL  ->  TION          conditional    ->  condition
                                    rational       ->  rational
    (m>0) ENCI    ->  ENCE          valenci        ->  valence
    (m>0) ANCI    ->  ANCE          hesitanci      ->  hesitance
    (m>0) IZER    ->  IZE           digitizer      ->  digitize
    (m>0) ABLI    ->  ABLE          conformabli    ->  conformable
    (m>0) ALLI    ->  AL            radicalli      ->  radical
    (m>0) ENTLI   ->  ENT           differentli    ->  different
    (m>0) ELI     ->  E             vileli       - >  vile
    (m>0) OUSLI   ->  OUS           analogousli    ->  analogous
    (m>0) IZATION ->  IZE           vietnamization ->  vietnamize
```

```
     (m>0) IZATION ->  IZE           vietnamization ->  vietnamize
     (m>0) ATION   ->  ATE           predication    ->  predicate
     (m>0) ATOR    ->  ATE           operator       ->  operate
     (m>0) ALISM   ->  AL            feudalism      ->  feudal
     (m>0) IVENESS ->  IVE           decisiveness   ->  decisive
     (m>0) FULNESS ->  FUL           hopefulness    ->  hopeful
     (m>0) OUSNESS ->  OUS           callousness    ->  callous
     (m>0) ALITI   ->  AL            formaliti      ->  formal
     (m>0) IVITI   ->  IVE           sensitiviti    ->  sensitive
     (m>0) BILITI  ->  BLE           sensibiliti    ->  sensible

Step 3

     (m>0) ICATE ->  IC              triplicate     ->  triplic
     (m>0) ATIVE ->                  formative      ->  form
     (m>0) ALIZE ->  AL              formalize      ->  formal
     (m>0) ICITI ->  IC              electriciti    ->  electric
     (m>0) ICAL  ->  IC              electrical     ->  electric
     (m>0) FUL   ->                  hopeful        ->  hope
     (m>0) NESS  ->                  goodness       ->  good

Step 4
```

```
     (m>1) AL    ->                  revival        ->  reviv
     (m>1) ANCE  ->                  allowance      ->  allow
     (m>1) ENCE  ->                  inference      ->  infer
```

```
     (m>1) ER     ->                  airliner        ->  airlin
     (m>1) IC     ->                  gyroscopic      ->  gyroscop
     (m>1) ABLE   ->                  adjustable      ->  adjust
     (m>1) IBLE   ->                  defensible      ->  defens
     (m>1) ANT    ->                  irritant        ->  irrit
     (m>1) EMENT  ->                  replacement     ->  replac
     (m>1) MENT   ->                  adjustment      ->  adjust
     (m>1) ENT    ->                  dependent       ->  depend
     (m>1 and (*S or *T)) ION ->      adoption        ->  adopt
     (m>1) OU     ->                  homologou       ->  homolog
     (m>1) ISM    ->                  communism       ->  commun
     (m>1) ATE    ->                  activate        ->  activ
     (m>1) ITI    ->                  angulariti      ->  angular
     (m>1) OUS    ->                  homologous      ->  homolog
     (m>1) IVE    ->                  effective       ->  effect
     (m>1) IZE    ->                  bowdlerize      ->  bowdler

Step 5a

     (m>1) E      ->                  probate         ->  probat
                                      rate            ->  rate
     (m=1 and not *o) E ->            cease           ->  ceas

Step 5b

     (m > 1 and *d and *L) -> single letter
                                      controll        ->  control
                                      roll            ->  roll
```

CAS : qui consiste à des règles simples pour trouver la racine de mot par exemple

Sleeping ⟶ Sleepe « supprimer ing et remplacer par e

voila les regles de cas

Step No.	Rule No.	Process	Porter's algorithm	Context-aware stemmer
1	1	Add		US → US
	2	Add		CEED → CESS
	3	Add		EED → EED
	4	Modify	IES → I	IES → Y
	5	Add		IED → Y
	6	Modify	(*V*) ING →	(*V*) ING → E
	7	Delete	(*V*) Y → I	
	8	Add		ED → E
2	9	Modify	ANCI → ANCE	ANCY → ANCE
	10	Modify	ENCI → ENCE	ENCY → ENCY
	11	Modify	ABLI → ABLE	ABLY → ABLE
	12	Modify	OUSLI → OUS	OUSLY → OUS
	13	Modify	ALITI → AL	ALITY → AL
	14	Modify	IVITI → IVE	IVITY → IVE
	15	Modify	BILITI → BLE	BILITY → BLE
	16	Add		FULLY → FUL
	17	Add		FUL →
	18	Add		LESSLY → LESS
	19	Add		BLY → BLE
3	20	Add		LESS →
	21	Modify	ICITI → IC	ICITY → IC
4	22	Modify	AL →	AL → E
	23	Add		IABLE → Y
	24	Delete	ANCE →	
	25	Add		SCOPIC → SCOPE
	26	Delete	IC →	
	27	Delete	ATE →	
	28	Add		FYE → FY
	29	Add		ALLY → AL
	30	Add		TLY → T
5	31	Delete	E →	

SECAS : L'algorithme SECAS a été proposé pour atténuer les problèmes de l'approche traditionnelle de radicalisation qui effectuent une transformation en aveugle de tous les termes de requête et de document sans tenir compte du contexte du mot racine pour une recherche efficace. Le flux d'application implique l'enrichissement de l'algorithme CAS, en ajoutant une phase de dictionnaire pour assurer la génération de mots racines significatifs. La méthodologie proposée comprend certaines étapes de prétraitement qui sont la tokénisation, la suppression des chiffres, les ponctuations et la suppression des mots vides avant d'entrer dans le processus d'indexation. En outre, il est utilisé la base de données Wordnet pour obtenir les racines les plus précises en fonction de la nature des mots d'origine (nom, verbe, adverbe ou adjectif). L'identification de la racine avec le Wordnet est sur la base du calcul de la mesure de similarité sémantique, qui s'est avérée efficace dans le cas du langage naturel.

Le tableau suivant montre la différence entre 3 les algorithme :

Tableau 2 :la différence entre cas ,porter, secas

PORTER	CAS	SECAS	
supprimer les suffixes et les préfixe	consiste à des règles simples	Combinent entre l'algorithme CAS et le dictionnaire WORDNET	Comment trouver la racine
NON	NON	OUI En utilisant les définitions, les synonymes et les contraires trouvés dans Wordnet	Conservez-vous le sens du mot une fois converti à la racine
Ne donne aucune importance à la signification du mot	deux mots ayant le même sens ne peut être réduit à la même racine, deux mots avec des significations distinctes ne peuvent pas être sépares		Les problèmes qui existent

3.3.1.1 Traitement de demande

Le processus d'indexation commence par une phase d'acquisition du document, suivie d'une phase de prétraitement; qui comprend la tokénisation, la suppression des mots vides…etc. Le processus d'indexation inclut à la fois un algorithme d'indexation basé sur les règles de l'algorithme Porter avec une version modifiée de l'algorithme CAS (ajout de quelques étapes de traitement) et une phase d'indexation basé dictionnaire; utilisant l'ontologie Wordnet. Les résultats obtenus sont comparés et enfin le SECAS L'algorithme stemming est appliqué pour obtenir les termes de l'index final (comme illustré à la figure 4.2).

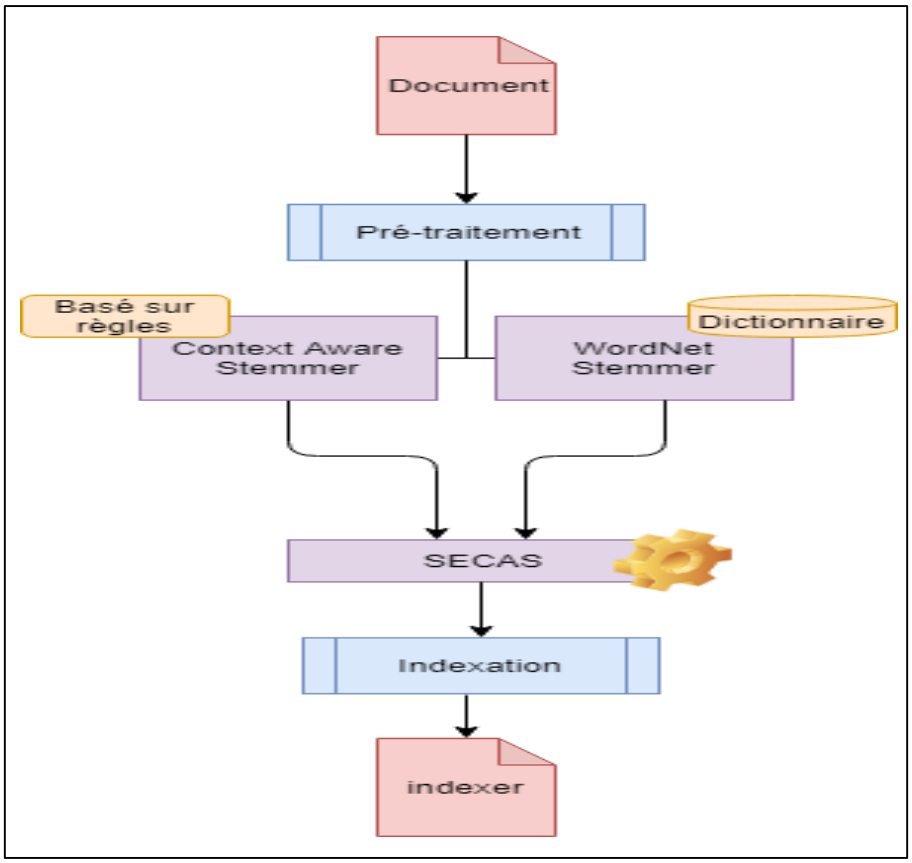


Figure 11: Diagramme de la méthode d'indexation.

Cette architecture peut être décomposée en 7 étapes:

- Étape 1: Le document texte est donné en entrée dans le stemmer.
- Étape 2: Suppression de la ponctuation, des chiffres et des mots vides comme préposition, conjonction, article, etc.
- Étape 3: Identification des descripteurs CAS.
- Étape 4: Identification des descripteurs Wordnet.
- Étape 5: Application de l'algorithme SECAS et comparaison des descripteurs.
- Étape 6: Identification des descripteurs les plus pertinents.
- Étape 7: Création du fichier d'index final en tant que sortie du stemmer.

L'approche de suppression des affixes accompagnée d'un dictionnaire algorithme contribuera à la réduction de l'espace occupé par un mot dans la mémoire de l'ordinateur; principalement parce que dans les langues naturelles, il y a beaucoup de mots qui diffèrent dans la façon dont

ils sont écrits mais partagent des racines similaires. Dans les sous-sections suivantes, nous décrivons les différents modules présentés dans le diagramme ci-dessus.

3.3.1.2 Module de traitement du langage naturel

Avant que les documents ne soient traités, le prétraitement des données techniques sont appliquées sur la collection afin de réduire sa taille en supprimant autant de variantes structurelles de mots ayant la même signification que possible. Cette action augmenterait l'efficacité du système RI. Le prétraitement est l'une des phases les plus importantes dans nombreux processus de traitement de texte.

Notre phase de prétraitement comprend:

- Tokenization: «La tokenisation est le processus de décomposition des phrases ainsi que le fichier texte en mot délimité par un espace, une tabulation ou une nouvelle ligne". En d'autres termes, la tokenization est le processus de conversion d'un flux de caractères (le texte des documents) dans un flux de mots (les mots candidats à adopter comme termes d'index) c.-à-d. l'identification des différents mots du texte [29].
- Suppression des mots vides: "Un mot vide (stop word en Anglais) est un mot qui n'a pas de sens» [30]. Les mots les plus courants représentent environ 50% du contenu du texte dans la plupart des langues [31]. Ces mots ne sont pas utiles pour décrire le besoin d'information de l'utilisateur. C'est la même chose pour les déterminants, conjonctions de coordination, prépositions, articles, verbes auxiliaires, pronoms relatifs, etc. Ainsi, ces mots ne font que ralentir l'IRS sans améliorer la qualité des résultats [32]. La suppression des mots vides et des doublons est une fonction que le algorithme actuel de Porter ainsi que l'algorithme CAS ne traite pas. Cependant, dans le cas de l'algorithme SECAS, nous inclus une phase d'élimination des mots vides.
- Élimination de la ponctuation: pour obtenir de bons résultats et accélérer le processus d'indexation. Il est également important de supprimer toutes les ponctuations ainsi que les accents inclus dans le contenu des textes au fur et à mesure car ils ne sont pas de sens et ne sont pas pertinents pour une tâche de recherche donnée.

3.3.1.3 Module d'indexation

Une fois toutes les étapes mentionnées ci-dessus effectuées, les mots du document sont prêts pour le processus de stemming. Avec l'algorithme SECAS, nous voulions proposer une technique d'indexation combinant des fonctionnalités issues des stemmers basés dictionnaire et ceux à base de règles.

Algorithm 1: The Semantically Enriched Context-Aware Stemming Algorithm

```
1  SECAS (word)
   Input  : word as a string
   Output: indexTerm a string representing the best index term
2  begin
3      wordnetIndexTerm ⟵ WordnetStemming(word);
4      casIndexTerm ⟵ CAS(word);
5      indexTerm: empty string;
6      if word is a compound noun then
7          indexTerm ⟵ word;
8      end
9      else if wordnetIndexTerm <> word OR casIndexTerm <> word
        then
10         if length(wordnetIndexTerm) < length(casIndexTerm) AND
            length(wordnetIndexTerm) > 0 then
11             indexTerm ⟵ wordnetIndexTerm;
12         end
13         else if casIndexTerm has at least one definition then
            casIndexTerm is shorter than wordnetIndexTerm
14             indexTerm ⟵ casIndexTerm;
15         end
16     end
18     return indexTerm
19 end
```

Algorithm 2: Wordnet Stemming

```
1  WordnetStemming(word)
   Input  : word as a string
   Output: stem the shortest Wordnet stem as a string
2  begin
3      stem: string;
4      noun: shortest stem whose type is noun;
5      verb: shortest stem whose type is verb;
6      adjective: shortest stem whose type is adjective;
7      adverb: shortest stem whose type is adverb;
8      set nounPriority = 1 AND verbPriority = 2 AND
       adjectivePriority = 3 AND adverbePriority = 4 ;
       /* 1 is the highest priority.  Nouns being more
       significant than verbs, verbs more significant than
       adjectives, Etc.                                    */
9      stem ⟵ shortest meaningful stem with the highest priority;
11     return stem
12 end
```

3.3.2 Module de calcul de similarité

La mesure que nous avons adopté, se base sur une combinaison pondérée d'une mesure de similarité à fiable niveau, c'est-à-dire similarité terminologique (elle-même, issue de la combinaison d'une similarité lexicale et une similarité syntaxique) et similarité structurelle. L'algorithme de similarité renvoi une valeur comprise entre 0 et 1. Tout d'abord, les mots-clés de la requête et les index de documents sont représentés par des vecteurs, où chaque terme d'une requête est comparé respectivement avec tous les termes indexes. Enfin, la similarité d'une requête avec l'ensemble les documents du corpus est calculée. Plus précisément, la similarité entre un mot-clé de requête et l'index de document est calculée à partir d'une matrice de corrélation sémantique. Où les lignes représentent les termes de la requête et les colonnes représentent les termes du document indexé. Une explication détaillée est donnée ci-dessous

3.3.2.1 Similarité terminologique

Mesures de similarité terminologique est calculée sur la base d'une comparaison syntaxique et lexicale.

3.3.2.2 Similarité syntaxique

Ces méthodes sont basées sur la comparaison de mots, de chaines de caractère ou de texte basés sur les lettres qu'ils ont en commun. Nous utilisons la distance Jaro. La distance Jaro mesure la similarité entre deux chaines de caractères.

Cette mesure est particulièrement adaptée à la comparaison des chaines courtes et sera donc parfaite pour mapping les description de documents et les mots-clés des requêtes. Le résultat est normalisé de façon à avoir une mesure entre 0 et 1 , zéro représente l'absence de similarité et 1 l'Egalite des chaines comparées. La distance **Jaro** entre deux chaine de caractère S1 et S2 est défini comme suit [4]:

$$synSim = \frac{1}{3}\left(\frac{m}{|S_1|} + \frac{m}{|S_2|} + \frac{m-t}{m}\right)$$

Où : m est le nombre de caractères correspondants.

t est le nombre de transposition.

$|S_i|$ est la longueur de la chaine de caractère S_i.

Deux caractères identiques de S1 et S2 sont considérés comme correspondants si leur éloignement (i.e.la différence entre leurs positions dans leurs chaines respectives) ne dépasse pas [4]:

$$\left(\frac{\max(|S_1|,|S_2|)}{2}\right)-1$$

Exemple

Soient deux chaines S1 MARTHA et S2 MARHTA. Nous allons dresser leur matrice de correspondance. Ici, éloignement maximal est égal à 2. Dans les cases de la matrice ci-dessous, on inscrira donc 1 lorsque les caractères sont identique et 0 sinon :

	M	A	R	T	H	A
M	1	0	0	0	0	0
A	0	1	0	0	0	0
R	0	0	1	0	0	0
H	0	0	0	0	1	0
T	0	0	0	1	0	0
A	0	0	0	0	0	1

m = 6 (nombre de 1 dans la matrice), $|S1| = 6$, $|S2| = 6$.

Les caractères correspondants sont {M A R T H A } pour S1 et {M A R H T A} pour S2. En considérant ces ensembles ordonnés, on a donc 2 couples (T/H et H/T) de caractères correspondants différents, soit deux demi-transposition. D'où T=2/2=1

La similarité syntaxique est donc calculée:

SynSim=1/3(6/6 + 6/6 + 6-1/6) =0.944

3.3.2.3 Similarité lexicale

Les méthodes lexicales nécessitent l'utilisation de ressources externes. Plusieurs types de ressources peuvent être utilisés, mais nous avons choisi WordNet.

Wordnet est une grande base de données lexicale pour la langue anglaise où les noms, les verbes, les adjectifs et les adverbes sont regroupés dans des ensembles de synonymes cognitifs (appelés synsets). Les synsets sont liés par moyens de relations conceptuelles sémantiques et lexicales.

La formule de similarité lexicale entre S1 et S2 se calcule ainsi [4]:

$$lexSim(S_1, S_2) = \frac{\beta}{\min(|Syn(S_1)|, |Syn(S_2)|)}$$

Où : min (|syn(S1)|, (|syn(S2)|) le minimum des cardinalités de deux ensembles |syn(C1)| et |syn(C1)| β=(|syn(S1)∩ syn(S2)|.

Cette mesure renvoi 1 si au moins S1 et S2 ont 1 synset commun. 0 est retournée dans le cas ou S1 et S2 ne sont pas synonymes et n'ont pas de relation lexicale (antonymes, hyponymes…).

Après le calcul de la similarité lexicale et syntaxique, les valeurs sont combinées par la similarité terminologique avec la formule suivante [4]:

$$terSim(S_1, S_2) = \frac{(lexSim(S_1, S_2) \times lexCoeff) + (synSim(S_1, S_2) \times synCoeff)}{(lexCoeff + synCoeff)}$$

Ou Coeff est un coefficient numérique calculé comme suit : Coeff=exp^{sim}

3.3.2.4 Similarité structurelle

Les méthodes de similarité structurelle déduisent la similarité de deux mots, en utilisant des informations structurelles, les méthodes calculent la similarité entre deux concepts (mots, chaînes de caractère.) en utilisant soit des informations sur leur structure interne, ou bien externe utilisant la structure hiérarchique d'une ontologie. Et ce, en comptant le nombre d'arcs dans la hiérarchie pour déterminer la similarité entre deux entités. La similarité de **Wu et Palmer** est une mesure qui utilise également la notion de Plus Court Chemin (PCC) entre les concepts mais dépend aussi de leur position dans le réseau. En effet, la mesure de Wu et Palmer a tendance à exprimer un éloignement sémantique pour des concepts proches de la racine. Ainsi pour deux concepts C1 et C2, Wu et Palmer calcule leur similarité de la manière suivante :

$$\text{Sim(c1,c2)} = \frac{2 * PCC(LCS(C1,C2),RACINE)}{PCC(LCS(C1,C2),C1)+PCC(LCS(C1,C2),C2)+2*PCC(LCS(C1,C2),RACINE)}$$

Où PCC(c1,c2) est Le plus court chemin de deux concepts est le nombre minimal d'arêtes pour aller d'un concept à un autre.

Le LCS (Least Common Subsumer) de deux concepts est le concept hyperonyme à ces deux concepts.

3.3.2.5. Similarité sémantique

Après le calcul des similarités terminologiques (comparaison syntaxique et lexicale) et similarité structurelle, la similarité sémantique est calculée par la combinaison des deux grâce à la formule :

$$semSim(S_1, S_2) = \frac{(terSim(S_1, S_2) \times terCoeff) + (strSim(S_1, S_2) \times strCoeff)}{(terCoeff + strCoeff)}$$

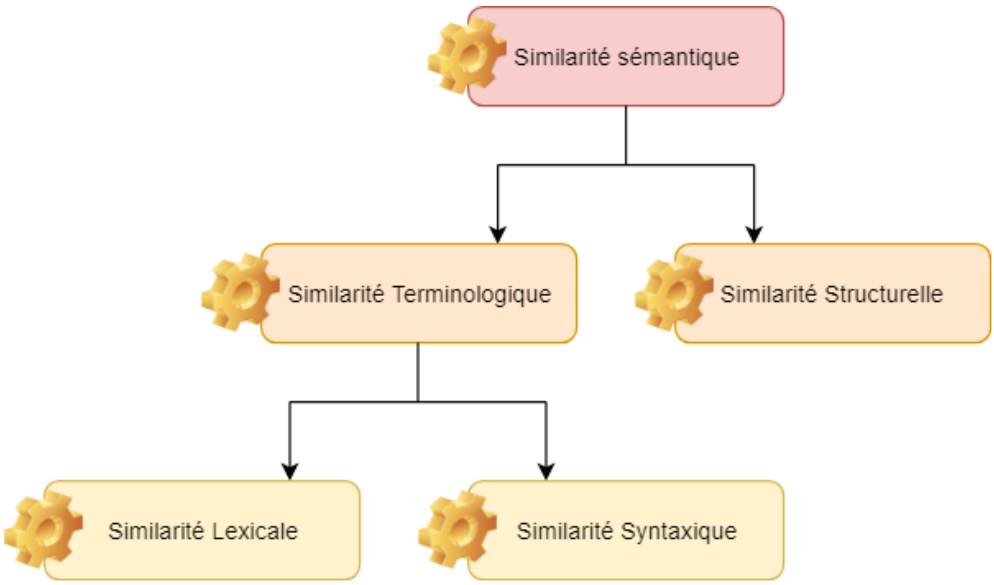


Figure 12: Diagramme récapitulatif de la mesure sémantique utilisée.

Après ces étapes, nous obtenons une matrice de corrélation, où les cellules contiennent des mesures de similarité. Soit M la matrice de corrélation, où chaque élément(i , j) décrit la similarité sémantique semSim(qi , tj) entre un mots clés de la requête qi et les termes de document indexe tj. A partir de cette matrice, un vecteur de similarité simVecrtor est généré. Tel que la longueur du vecteur est égale au nombre de mots-clés de la requête et chaque élément Vi du vecteur représente la MAX(semSim(qi, tj)) de la ligne correspondante dans la matrice. Cette étape garantit que seule la valeur de similarité la plus élevée est conservée pour les mots-

clés de la requête avec un document donné. Enfin, afin d'obtenir le mapping document-requête « simDegree », nous procédons par le calcul de la similarité moyenne. Tel que [4] :

$$\text{AVGsim(simVector)}=\frac{\sum vi}{|simVector|}$$

$$qi\overbrace{\begin{bmatrix} M1,1 & \dots & M1,n \\ \vdots & \ddots & \vdots \\ Mm,1 & \dots & Mm,n \end{bmatrix}}^{dj} \rightarrow \overbrace{\begin{bmatrix} v1 \\ \vdots \\ vm \end{bmatrix}}^{\max(semSim(qi,tj))} \rightarrow simDegree = \quad VGSim(simVector)$$

La figure ci-après illustre le SRI que nous avons développé pour tester notre algorithme de stemming.

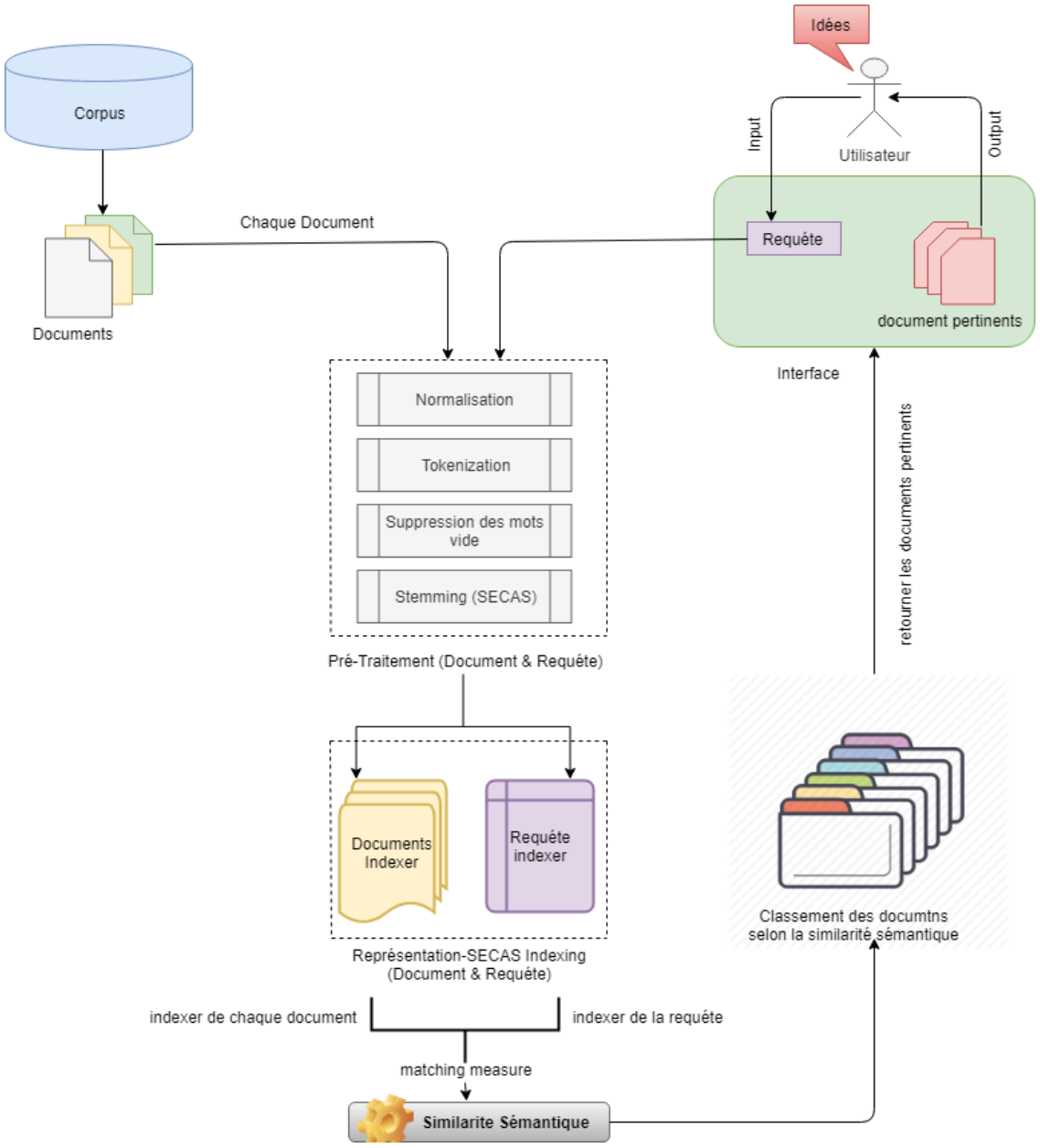


Figure 13 : Architecture globale du SRI utilisé.

3.4 Conclusion

Dans ce chapitre, nous avons présenté le diagramme de notre contribution à un SRI sensible au contexte. Les algorithmes de Porter et de CAS ont été étudiées avec pour but de proposer une nouvelle méthode hybride d'indexation basé sur l'extraction de racines significatives. Le principal avantage de notre méthode par rapport aux méthodes existantes est qu'elle fournit un stemmer précis avec des racines significatives dans 99% des cas.

Implémentation

Chapitre4 :Implementation

4.1 Introduction

Dans ce chapitre, nous allons présenter l'implémentation de notre système. Nous commençons tout d'abord par la présentation des langages de programmation et environnements de développement, en détaillant les différents outils utilisés dans chaque étape, à commencer par la représentation des données et se terminant par le retour des informations pertinentes à l'utilisateur. L'étape intermédiaire comprend les opérations d'indexation, de filtrage, de recherche, de mise en correspondance et de classement. Puis nous expliquons le déroulement de l'application, et enfin nous interprétons et commentons les résultats obtenus.

4.2 Environnement de développement

Nous présentons dans cette section, le langage de programmation Python utilisé et son environnement de développement PyCharm.

4.2.1 Python

Python est un langage de programmation interprété de haut niveau qui a été initialement conçu par Guido van Rossum à la fin des années 1980 en tant que membre de l'Institut national de recherche en mathématiques et en informatique. Bien entendu, Python, à l'instar d'autres langages, a connu plusieurs versions. Python 0.9.0 a été publié pour la première fois en 1991. Outre la gestion des exceptions, Python incluait des classes, des listes et des chaînes.

En 2000, Python 2.0 a été publié. Cette version était plutôt un projet à source ouverte émanant de membres de l'Institut national de recherche en mathématiques et en informatique. Cette version de Python incluait des connaissances de liste, un collecteur d'ordures complet et la prise en charge de l'Unicode.

Python 3.0 était la version suivante et a été publié en décembre 2008 (la dernière version de Python est 3.8). Bien que Python 2 et 3 soient similaires, il existe des différences subtiles. Le fonctionnement de l'instruction « print » est peut-être celui qui convient le mieux. Comme dans Python 3.0, l'instruction « print » a été remplacée par une fonction « print () » [46].

Python fournit de nombreuses fonctionnalités faciles à apprendre et à utiliser. C'est un langage plus expressif, ce qui signifie qu'il est plus compréhensible et lisible comparé à d'autres langages.

- Python est un langage interprété, c'est-à-dire que l'interprète exécute le code ligne par ligne à la fois. Cela facilite le debugging et convient donc aux débutants.
- Il est multi plates-formes : les programmes tournent sans modification sur tous les environnements où Python existe (Windows, Unix et Mac).

4.2.2 PyCharm

PyCharm [48], est un environnement de développement intégré (EDI), utilisé en programmation informatique, spécialement pour le langage Python. Il est développé par la société tchèque JetBrains5. Il fournit une analyse de code, un débogueur graphique. PyCharm est multiplateforme, supporté sur les versions de Windows, MacOs et Linux. L'édition communautaire est publiée sous la licence Apache.

Il existe également une édition professionnelle avec des fonctionnalités supplémentaires - publiée sous une licence propriétaire.

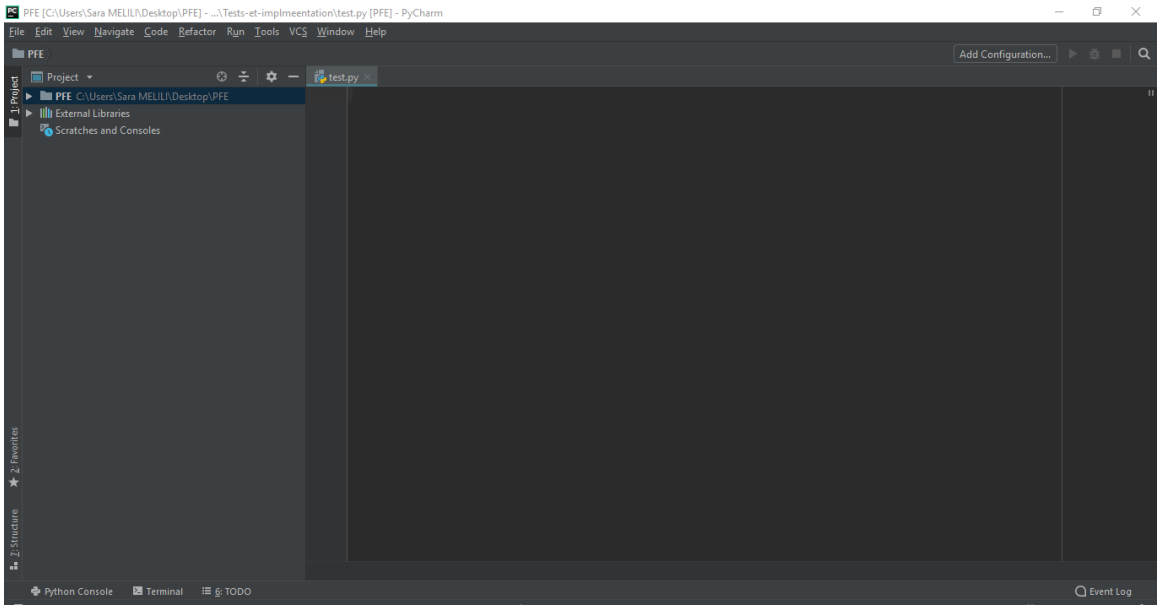


Figure 14: Environnement PyCharm.

4.3 Dataset cranfield

Les expériences de Cranfield étaient des expériences de recherche d'informations sur ordinateur menées par Cyril W. Cleverdon au College of Aeronautics de Cranfield dans les années 1960, afin d'évaluer l'efficacité des systèmes d'indexation.

Ils représentent le modèle d'évaluation type des systèmes de récupération d'informations, et ce modèle a été utilisé dans le cadre d'efforts d'évaluation de récupération d'informations à grande échelle tels que la conférence de récupération de texte (TREC)[2]. Le modèle d'évaluation repose sur trois composants:

1. Une collection de documents (ou corpus) « cran.all »,
2. Un ensemble de requêtes « cran. qrey »,
3. Un ensemble de jugements de pertinence « cranqrel », c'est-à-dire un fichier qui, pour chaque requête, répertorie les documents considérés comme pertinents pour répondre aux questions données (requêtes).

4.4 Description du Système

Notre système contient trois packages principaux qui sont : prétraitement (normalisation, tokenization, suppression des mots vides), indexation(documents et requêtes), calcule de similarité.

4.4.1 Prétraitement

Dans cette étape, nous avons implémenté notre propre algorithme avec python et nous avons utilisé différentes bibliothèques pour chaque sous-tâche.

4.4.1.1 Normalisation

C'est un processus qui convertit une liste de mots en une séquence plus uniforme. Par exemple il traite la capitalisation (ChAt→ chat) et supprime les accents et les signes diacritiques

Exemple : résumé →resume, naïve → naive

4.4.1.2 Tokenization

C'est le processus consistant à découper les espaces blancs et à supprimer les caractères de ponctuation. Chaque terme résultant de cette tâche est appelé « Token ».

[2] https://trec.nist.gov/

Input : “ Bonjour monsieur, est ce que le directeur est là ?. “

Output : |Bonjour| |monsieur| |est| |ce| |que| |le| |directeur| |est| |là|.

4.4.1.3 Suppression des mots vide

Dans cette étape nous allons supprimer tous les mots vides (the, and …) et les mots qui n’affectent pas les résultats de recherche (about, best, also…)

4.4.1.4 Stemming

Dans cette étape, nous avons utilisés 3 algorithmes pour obtenir un bonne stem (SECAS, CAS, et Porter). Le tableau 2 représente des exemples de résultats du stemming effectué avec CAS, SECAS, et Porter concernant les mots (sleeping, tokenization, experimental).

Tableau 3: Exemple de Stemming avec les 3 méthodes

	Porter	SECAS	CAS
Sleeping	sleep	sleep	sleepe
Tokenization	token	token	token
Experimental	experiment	experimental	experimente

Après le prétraitement des documents et requêtes, nous allons procéder par leur indexation.

4.4.2 Indexation

Dans cette étape, nous avons choisi une représentation des documents et des requêtes sous forme de contenu résumé avec porter et secas et cas.

L’indexation est une étape très importante dans le processus de RI. Elle consiste à déterminer et extraire les termes représentatifs du contenu d’un document ou d’une requête, qui couvrent au mieux leur contenu sémantique. La qualité de la recherche dépend en grande partie de la qualité de l’indexation.

L'indexation est un processus consistant à reformuler le contenu d'un document sous une forme plus adaptée à son exploitation dans une application donnée

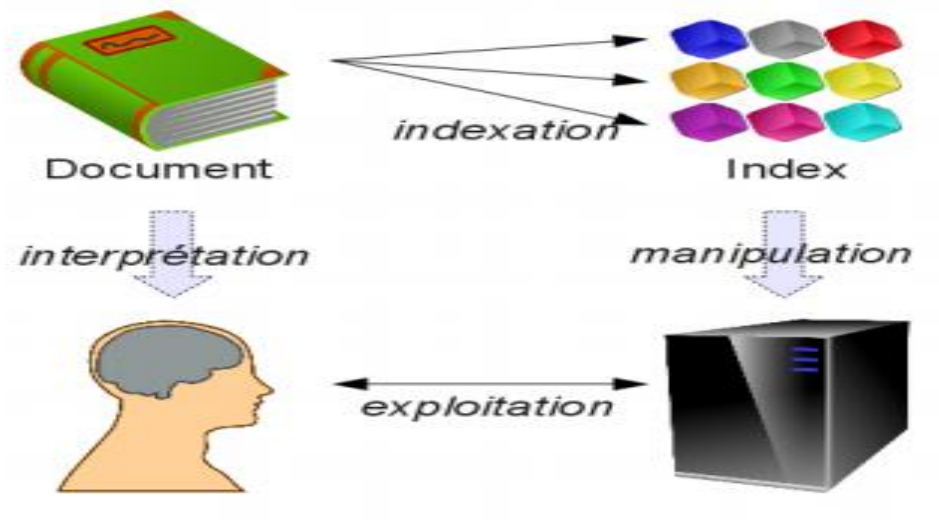


Figure 15:processus de l'indexation

Chaque mot d'un document ou d'une requête obtenu après le prétraitement passe par une indexation avec les algorithmes : SECAS, CAS et Porter. Pour obtenir sa forme de base (racine). La figure suivante montre le résultat d'indexation d'un document:

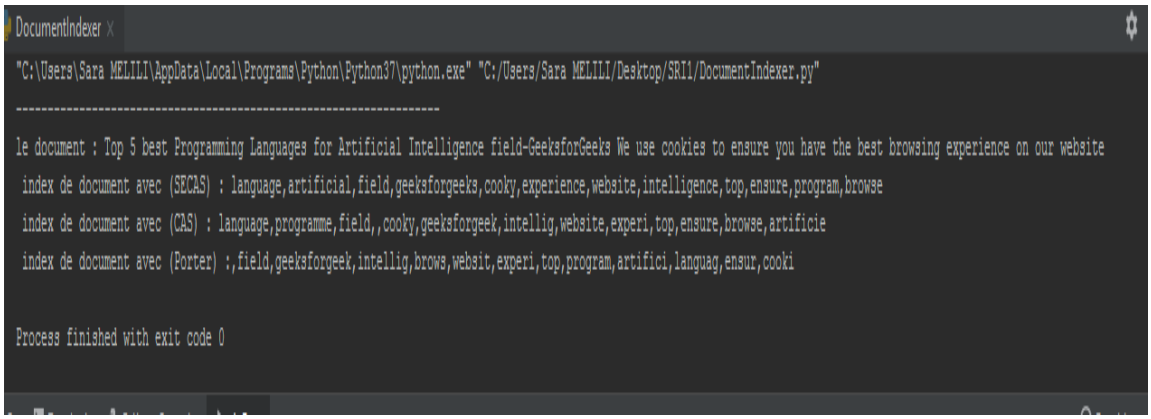


Figure 16: Résultat d'indexation d'un document.

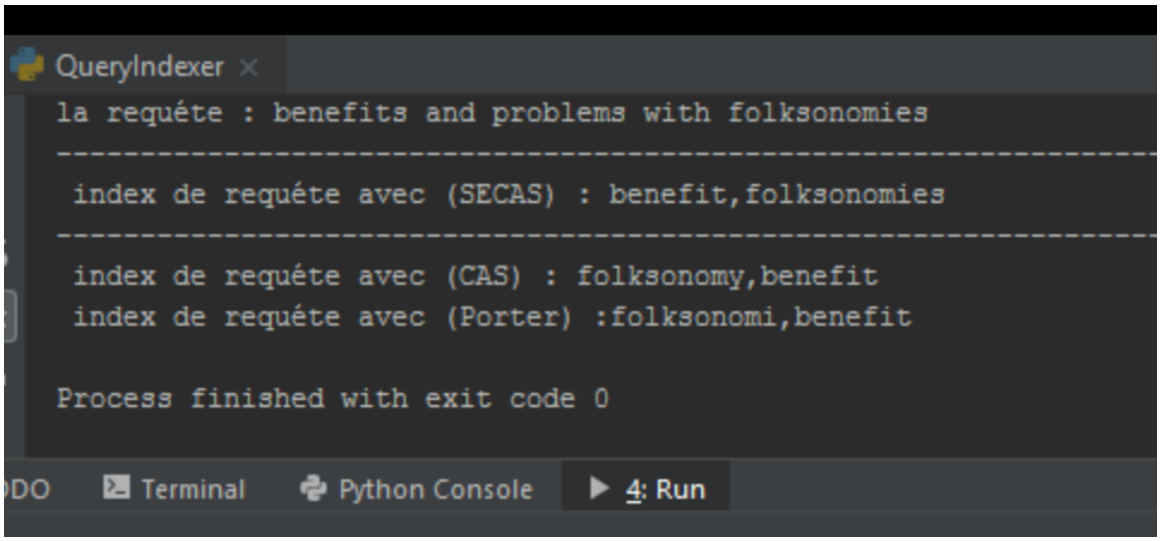


Figure 17:Résultat d'indexation d'une requête.

Le résultat de l'indexation est une liste de descripteurs du document ou de la requête. Ce dernier est souvent une liste de termes significatifs pour l'unité textuelle correspondante.

4.4.3 Similarité sémantique

Après avoir indexer chaque document et requête, nous devons calculer les appariements document-requête basé sur la similarité sémantique pour obtenir la ou les documents pertinents.

La similarité sémantique est calculée par la combinaison terminologique (comparaison syntaxique et lexicale) et structurelle.

A titre d'exemple, le tableau 3 représente les différentes similarités entre le mot job et work.

Tableau 4:Exemple de calcule la similarité.

Similarité Syntaxique	Similarité lexicale	Similarité structurelle	Similarité terminologique	Similarité sémantique
0.52	0.0	0.85	0.33	0.66

Une extrait de la similarité sémantique après indexation des documents et requêtes du corpus avec indexation SECAS est présenté dans la figure suivante:

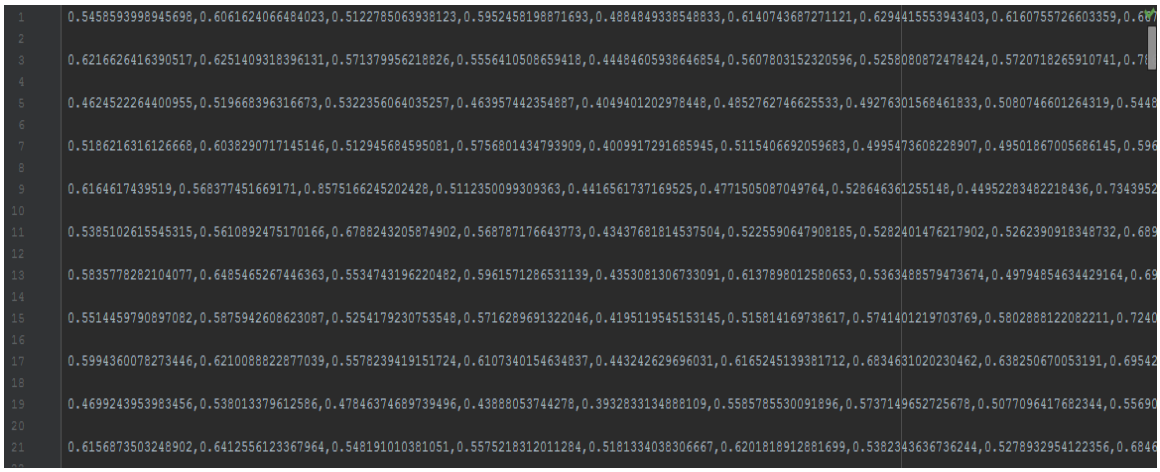


Figure 18: Extrait du calcul de similarité sémantique avec SECAS

Une extrait de la similarité sémantique après indexation des documents et requêtes du corpus avec indexation Porter est présenté dans la figure suivante:

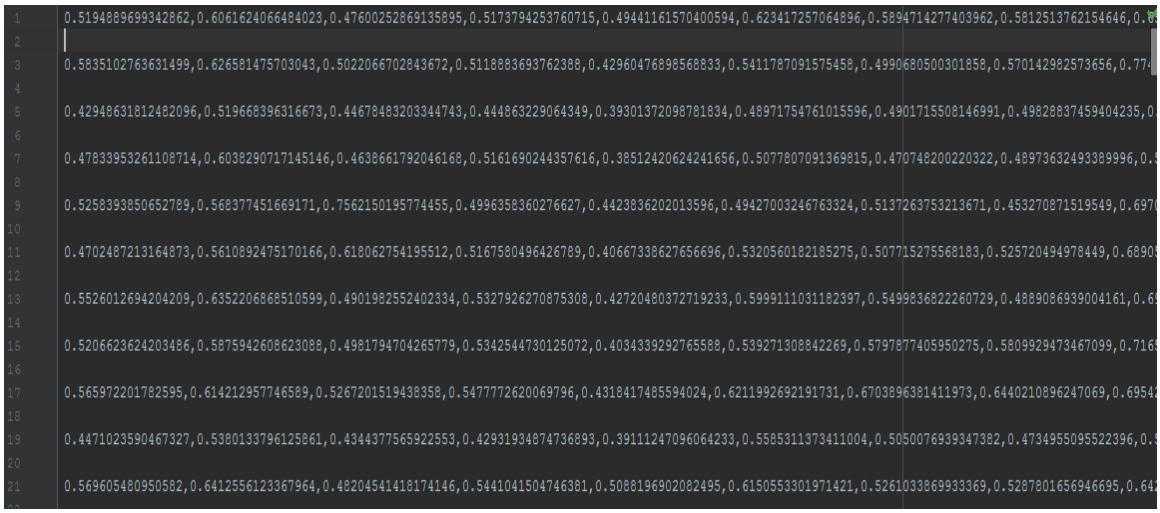


Figure 19: Extrait du calcul de la similarité sémantique avec Porter

Une extrait de la similarité sémantique après indexation des documents et requêtes du corpus avec indexation CAS est présenté dans la figure suivante:

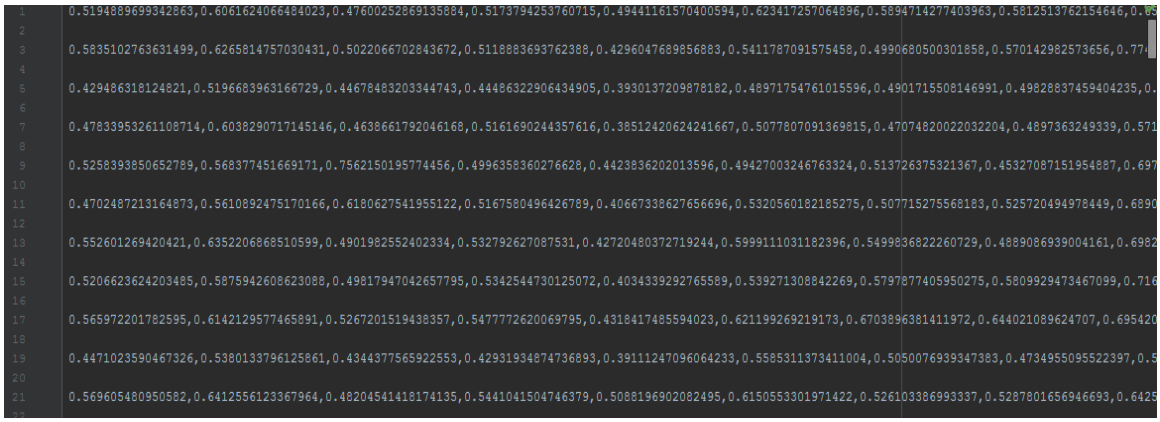


Figure 20: Extrait du calcul de la similarité sémantique avec CAS.

4.5 Evaluation et résultats

Pour indexer la collection cran.tar (avec 1399 documents et 255 requêtes), le temps d'indexation que nous avons obtenu avec SECAS prend deux fois plus de temps que celui de CAS ou PORTER. De même que pour le calcul des similarité après indexation. Nous pensons que ceci est dû au fait que SECAS comprends une double technique d'extraction des racines et lors de la phase de calcul de similarité, vue que les indexeurs possèdent tous des sens, le calcul de similarité global est plus gourmand en temps. Le tableau 4 ci-dessou récapitule les différents temps d'exécution obtenus.

Tableau 5: Temps d'exécution des différents algorithmes avec le dataset.

	Algorithme	**Millisecondes**	**Minutes**
Indexation des requêtes	Porter	25	0.00004
	Secas	50	0.00008
	Cas	30	0.00005
Indexation des documents	porter	293	0.0004
	secas	398	0.0007
	cas	391	0.0006
Calcul des appariements documents-requêtes	porter	77974	0.12
	Secas	126441.4	0.21
	Cas	114000	0.19

Pour évaluer les différents algorithmes, nous avons calculer les mesures de performances notamment : rappel, précision ,f-mesure, et accurancy.

Le rappel est défini par le nombre de document pertinents retrouvés au regard du nombre de documents pertinents. Le rappel est donc calculé comme suit :

$$R = \frac{\text{vrai\_pos}}{\text{vrai\_pos} + \text{faux\_neg}}$$

La précision est le nombre de documents pertinents retrouvés rapporté au nombre de documents total proposés pour une requête donnée. La précision est donc calculée comme suit :

$$P = \frac{\text{vrai\_pos}}{\text{vrai\_pos} + \text{faux\_pos}}$$

F-mesure : une mesure qui combine la précision et le rappel est leur moyenne harmonique. Elle se calcule comme suit :

$$F = 2.\frac{(\text{P.R})}{(\text{P} + \text{R})}$$

Accuracy : dans la mesure d'un ensemble, accuracy fait référence à la proximité des mesures à une valeur spécifique. L'accuracy peut nous dire si un modèle est correctement formé, et elle se calcule comme suit :

$$\text{Acc} = \frac{\text{vrai\_pos} + \text{vrai\_neg}}{vrai_{pos} + vrai_{neg} + faux_{pos} + faux\_neg}$$

Où :

Vrai_pos : le résultat du test et la réalité sont égaux. C'est-à-dire que le SRI indique un résultat positif alors que le fait étudié correspond un cas positif.

Vrai_neg : le résultat du test est contraire à la réalité. C'est-à-dire que le SRI indique un résultat positif alors que le fait étudié correspond un cas négatif.

Faux_pos : le résultat du test et la réalité sont égaux. C'est-à-dire que le SRI indique un résultat négatif alors que le fait étudié correspond un cas négatif.

Faux_neg : le résultat de test est contraire à la réalité. C'est-à-dire que le SRI indique un résultat négatif alors que le fait étudié correspond un cas positif.

Tableau 6: Matrice de confusion

	Réalité positif	Réalité négatif
Résultat de test positif	Vrai_pos	Faux_pos
Résultat de test negatif	Faux_neg	Vrai_neg

Après les calculs des vrai_pos, vrai_neg, faux_neg et faux_pos, nous passons au calcul des valeurs de perfrmance.

Le tableau suivant indique les résultats obtenus

Tableau 7: Mesures de performance

	SECAS	**Porter**	**CAS**
Rappel	0.52	0.37	0.49
Précision	0.68	0.68	0.70
F-mesure	0.59	0.49	0.58
Accuracy	0.51	0.46	0.51

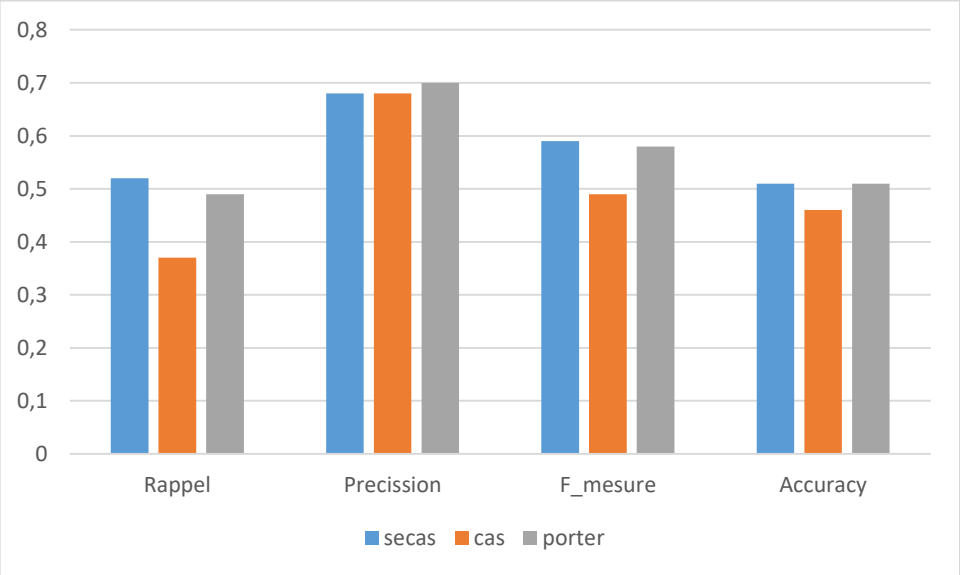


Figure 21: Mesures de performance du système.

4.5.1 Interprétation des résultats

Un système de recherche documentaire parfait fournira des réponses dont la précision et le rappel sont égaux à 1 (l'algorithme trouve la totalité des documents pertinents - rappel - et ne fait aucune erreur - précision). Dans la réalité, les algorithmes de recherche sont plus ou moins précis et plus ou moins pertinents. Il est possible d'avoir un algorithme avec un bon rappel. Dans notre cas, par exemple de algorithme SECAS a eu un score de précision de 0.68 avec un rappel de 0.52, ce qui signifie qu'il n'a trouvé que 52 % des réponses possibles. De même, qu'il

est possible de trouver des algorithmes très précis (par exemple algorithme CAS avec un rappel de 0,49, soit la quasi-totalité des documents pertinents et une précision 0.70).

4.6 Conclusion

Dans ce chapitre, nous avons décrit nos outils de développement ainsi que la collection de tests que nous avons utilisé. Par ailleurs, nous avons présenté nos résultats encourageants mais un peu biaisés.

Nous expliquons ces résultats, au fait que les valeurs des similarités documents-requêtes données dans la collection de CRANFIELD était catégorisés en cinque catégories : -1, 0, 1, 2, 3, 4, 5 avec une certaine confusion dans les catégories. Nos résultats étant des chiffres réel compris entre 0 et 1, il nous était difficile de convertir nos valeurs pour qu'elles correspondent aux catégories avant de passer au calcul de mesure de performance.

Notons que dans les cas limites, un système de recherche documentaire qui renvoie la totalité des documents de sa base aura un rappel de 1 mais une mauvaise précision, tandis qu'un système de recherche qui renvoie uniquement la requête de l'utilisateur aura une précision de 1 pour un rappel très faible. La valeur d'un SRI ne se réduit donc pas à un bon score en précision ou en rappel.

Toutefois, il est clair que d'autres tests mériteraient d'être conduits pour avoir une meilleure appréciation des résultats.

Conclusion et perspectives

Conclusion et perspectives

La recherche d'informations textuelles sur Internet peut être l'application la plus utilisée en Recherche d'Information et cette application dérive des techniques de Traitement Automatique de la Langue et de l'indexation et consiste à récupérer une grande quantité d'enregistrements de texte. La récupération des informations est réalisée à l'aide d'outils informatiques appelés système de recherche d'informations (SRI).

Avec le développement Web, la recherche d'informations a fait face à de nouveaux défis d'accès à l'information et ce, pour trouver des informations pertinentes dans un espace diversifié et de grande taille. Dans ce contexte, nous proposons une méthode d'indexation contextuelle riche en sémantique afin d'obtenir des résultats de recherche satisfaisants et compatibles avec la demande de l'utilisateur.

A ce titre, une méthode d'indexation et d'appariement entre requête et documents est proposée et évaluées. Cette méthode se concentre sur les aspects sémantiques des documents et des requêtes avec un algorithme de stemming (racinisation) sémantiquement enrichi basé sur l'algorithme connu de Porter.

L'algorithme de stemming commence par une phase de prétraitement, puis combine des fonctionnalités de stemmers algorithmiques et de stemmers basés dictionnaires visant à maximiser la proportion des racines significatives, sans compromettant les autres mesures de performance (c'est-à-dire améliorer le rappel sans diminuer la précision). En effet, la nouvelle méthode hybride de stemming est basé sur une combinaison de suppression d'affixes (basé sur Porter Stemming algorithme), des techniques contextuelles (basées sur la méthode contextuelle l'algorithme de Stemming "CAS"), et techniques basées sur corpus pour la langue Anglaise (avec la ressource Wordnet). L'algorithme de Stemming sémantiquement enrichi (SECAS) proposé peut être utilisé efficacement dans les étapes de traitement de la langue comme les systèmes de résumé et de classification du texte et aussi dans le cadre de la recherche d'information.

Les résultats obtenus sont bons, mais faute de temps et de moyens pour pouvoir faire des tests plus grands nous n'avions pas poussé nos tests afin de pouvoir apporter d'éventuelles améliorations aux algorithmes. Ainsi, comme perspectives d'enrichissement à notre travail, nous voudrions :

- Effectuer des tests avec des corpus plus grands comme le wt2g ou le wt10g afin de pouvoir comparer les résultats et les temps d'exécutions avec les versions initiales des algorithmes.
- Améliorer la complexité temporelle et spatiale des algorithme afin de les rendre moins gourmand et de ce fait plus adapté à des recherches en temps réel dans le cadre des réseaux sociaux par exemple.

Références bibliographies

[1] - Hernandez N. “Ontologie de domaine pour la modélisation du contexte en recherche d’information”, thèse de doctorat en informatique, Université Paul Sabatier. (2006)

[2] - Boubaker F. “Contribution à la définition de modèles de recherche d'information flexibles basés sur les CP-Nets”, thèse de doctorat en informatique, Université Paul Sabatier. 2008

[3] - Daoud M. “Accès personnalisé à l'information : approche basée sur l'utilisation d'un profil utilisateur sémantique dérivé d'une ontologie de domaines à travers l'historique des sessions de recherche”, thèse de doctorat en informatique, Université Paul Sabatier. (2009).

[4] - Mezzi, M. “Système de recherche sensible au contexte: Contribution a un Modelé sémantiquement enrichi pour la recherche d’information textuelle basée sur les folksonomies”, thèse de doctorat en informatique, Université de Blida 1. (2018).

[5] – Bouramoul, A. “ Recherche d’information Contextuelle et Sémantique sur le Web”, thèse de doctorat en informatique, Université MENTOURI de Constantine. (2011).

[6]- Anwar A. Alhenshiri, “Web Information Retrieval and Search Engines Techniques”, 2010, Al- Satil journal, PP: 55-92.

[8]- DjoerdHiemstra, “Information RetrievalModels”, published in Goker, A., and Davies,J. Information Retrieval: Searching in the 21stCentury. John Wiley and Sons, November2009, Ltd., ISBN-13: 978-0470027622.

[9]- Rich Brooks,Maine SEO. "What’s the Difference between a Directory and a Search Engine?".takeflyte.com 17 Juliet 2008.

[10]Schilit, B., Theimer, M. (1994). Disseminating active map informations to mobile hosts. In Institute of Electrical and Electronics Engineers (IEEE) Networks. V. 5, p. 22-32.

[11] Dey, K. A., Abowd, D. G. (2000). Towards a better understanding of context and context- awarness. In Computer Human Interactions (CHI2000) Workshop on the What, Who, Where and How of Context-Awarness.

[13]© Springer International Publishing AG 2018 P. Temdee and R. Prasad, Context-Aware Communication and Computing : Applications for Smart Environment, Springer Series in Wireless technology,DOI 10.1007/978-3-31959035-6_2

[14] Lieberman, H., Selcker, T. (2000). Out of context : Computer systems that adapt to, and learnfrom, context. In IBM System Journal. Juillet 2000.

[15] Mizouni, R., Matar, M. A., Al Mahmoud, Z., Alzahmi, S., & Salah, A. (2014). A framework for context-aware self-adaptive mobile applications SPL. Expert Systems with Applications, 41(16), 7549–7564.

[16] Chen, G., &Kotz, D. (2000). A survey of context-aware mobile computing research (Vol. 1,No. 2.1, pp. 2-1). Technical Report TR2000-381, Department of Computer Science, DartmouthCollege.

[17]Baldauf, M., Dustdar, S., & Rosenberg, F. (2007). A survey on context-aware systems.International Journal of Ad Hoc and Ubiquitous Computing, 2(4), 263–277.

[18]Schuster, S., Marhl, M., &Höfer, T. (2002). Modelling of simple and complex calciumoscillations.European Journal of Biochemistry, 269(5), 1333–1355.

[19] Prekop, P., & Burnett, M. (2003). Activities, context and ubiquitous computing. Computer Communications, 26(11), 1168–1176.

[20]Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., Altmann, J., &Retschitzegger, W.(2003,

January). Context-awareness on mobile devices-the hydrogen approach. In System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on IEEE (p. 10-pp).

[21]Liang, G., & Cao, J. (2015). Social context-aware middleware : A survey. Pervasive and Mobile Computing, 17, 207–219.

[22] Salber, D., Dey, K. A., Abowd, D. G., (1998). Ubiquitous Computing : Defining an HCIresearch agenda for an emerging interaction paradigm. In Georgia Tech GVU technical report.Janvier 1998.

[23] Brown, P. J. (1998). Triggering information by context. In personnel Technologies. V. 2, p. 1-9.

[24]] Alegre, U., Augusto, J. C., & Clark, T. (2016). Engineering context-aware systems and applications : A survey. Journal of Systems and Software, 117, 55–83.

[25] Winograd, T. (2001). Architectures for context. Human-Computer Interaction, 16(2), 401–419

[26] Dey, A. K. (2001). Understanding and using context. Personal and Ubiquitous Computing, 5(1),4–7..

[27]Chen, H. (2004). An intelligent broker architecture for pervasive context-aware systems. Baltimore County : University of Maryland.

[28] Siewiorek, D., Smailagid, A., Furukawa, J., Moraveji, N., Rieger, K., Shaffer, J. (2003).Sensay : A context-aware mobile phone. In proceedings of Symposium on wearable computers.

[29] Jayanthi, R. and C. Jeevitha (2015). An Approach for Effective Text Pre-Processing Using Improved Porters Stemming Algorithm. International Journal of Innovative Science, Engineering & Technology: 797-807.

[30] El-Defrawy, M., et al. (2015). CBAS: Context BASED ARABIC STEMMER. International Journal on Natural Language Computing: 1-12.

[31] Haven, o. (2016). "Text Tokenization and Processing in Text Indexes." Haven onDemand. Accessed June 2018, https://www.havenondemand.com.

[32] Singh Rajput, B. and N. Khare (2015). A survey of Stemming Algorithms for Information Retrieval. IOSR Journal of Computer Engineering: 76-80.

[34]. Studer, R., et al. (1998). "Knowledge Engineering: Principles and Methods." Data & Knowledge engineering 25(1-2): 161-197.

[35]. Widjaja, M. and S. Hansun (2015). Implementation of PORTER'S Modified Stemming Algorithm in an Indonisian Word Error Detection Plugin Application. International Journal of Technology: 139-150.

[36].Gormley, C. and Z. Tong (2014). Elastic search - The Definitive Guide. Accessed in June 2018, https://www.elastic.co.

[37].Hewlett Packard, E. (2015). "IDOL EXPERT." Vertica Advanced Analytics - myVertica.