RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SIENTIFIQUE

UNIVERSITÉ DE ABDELHAMID IBN BADIS DE MOSTAGANEM
Faculté des Sciences et Sciences de L'ingénieur
Département d'informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de master en informatique

Thème

Indexation Automatique de Documents Numérique

Présenté par:

Amine ROUKH
Abdelkadir SADOUKI

Encadré par:

Mme AIT SAADI

Année Universitaire: 2011-2012

Remerciements

Dieu merci de nous avoir donné le courage et la volonté pour la réalisation de ce travail.

En outre nous tenons à remercier notre cher encadreur Mme AIT SAADI pour son aide et ses judicieux conseils ainsi pour son entière disponibilité pour nous faciliter l'accomplissement de ce travail. Nous lui sommes infiniment reconnaissants de nous avoir encouragées et soutenu durant ce travail, nous avons ainsi pu apprécier sa rigueur scientifique, ses grandes qualités humaines et son œil critique qui nous a été très précieux pour structurer notre travail et améliorer sa qualité.

Nos vifs remerciements vont les membres de jury et le président qui nous avoir honorés de leur présence et d'avoir voulu évaluer ce travail.

Nous tenons à remercier ici tous ceux qui nous ont ouvert la voie dans cette année de douce folie et à tous ceux qui ont contribué de prés ou de loin à ce cheminement.

Amine & Abdelkadir

Dédicaces

Ce mémoire de master représente l'achèvement d'un long, fastidieux, mais oh combien enrichissant travail qui n'aurait pu voir le jour sans la participation, l'aide, les conseils, ou encore la présence de nombreuses personnes.

A ceux qui n'ont jamais cessé de nous apporter l'affection, l'amour, le courage et le bon sens, nos parents que dieu les protège.

A nos sœurs et frères qui ont toujours été présents à nos côtés.

A nos amis(es).

A nos professeurs.

Et à tous ceux qui nous aiment.

Table des matières

Liste des figures

Index des formules

Introduction générale

De nos jours, nous vivons dans une société de technologie et de progrès. L'être humain est assisté dans son quotidien par différentes formes de technologies dans l'accomplissement de ses tâches. L'information qui représente notre capital vital est également gérée et suivie par divers outils pour nous permettre de mieux l'utiliser. L'augmentation quasi-exponentielle des connaissances de l'homme ainsi que leur spécialisation inévitable dans des domaines d'intérêt très variés a conduit à la production d'un volume d'information sans précédant. La recherche d'information représente une passerelle entre l'utilisateur et cet univers d'informations qui ne cesse de croître.

Pour les utilisateurs il devient très difficile d'examiner et repérer ce qu'ils recherchent devant cette énorme masse d'informations disponible. Avoir des réponses pertinentes devient encore plus difficile, les systèmes de recherche d'information se veulent ainsi comme des messagers dont l'objectif principal est de répondre aux interrogations des utilisateurs. Cependant, comme tout système bâti sur un autre, ils doivent leur efficacité au processus d'indexation qui s'efforce de donner une représentation fidèle, compacte fiable et accessible de l'information recherchée.

L'indexation vient résoudre ce problème (de recherche d'information), dans ce contexte plusieurs méthodes d'indexation de documents ont été développées pour permettre de faire des représentations de documents de manière à pouvoir les repérer plus facilement et plus efficacement. Ces méthodes donnent de bons résultats au niveau du taux de rappel.

Pour améliorer le quotidien de tout un chacun en mettant à sa disposition l'information qu'il a souvent du mal à trouver, il faut faire en sorte que les systèmes de recherche d'informations soient plus efficaces et plus flexibles. Pour cela, l'indexation est l'un des axes auxquels, on devrait s'intéresser en premier lieu. Une des phases de l'indexation permet de stocker une abstraction des contenus des documents. Ces abstractions sont ensuite comparées à la représentation des besoins de l'utilisateur (la requête) à la phase d'interrogation (ou de recherche) grâce à une fonction de correspondance.

Pour répondre au mieux à ces besoins, nous avons été amenés à développer un système d'indexation automatique de documents numériques. On utilisera pour tester ce système le corpus

qui a été élaboré dans le cadre notre projet de licence « Réalisation d'un Crawler pour la recherche d'information Arabe sur le Web. Université de Mostaganem, 2010.

Avec l'élaboration de ce système nous avons rédigé un manuscrit que nous avons structuré comme suit :

Deux parties, la première présente un état de l'art qui est composée de deux chapitres 1et 2 :

Le chapitre 1 est un chapitre introductif qui donne les définitions de l'indexation, de documents et toutes les étapes de l'indexation, les différents types d'indexation leurs avantages et leurs limites.

Le chapitre 2 détaille le type d'indexation pour lequel on a opté : l'indexation automatique, il permet d'expliciter toute la démarche à suivre et toutes les méthodes utilisées pour faire une indexation automatique.

La deuxième partie de notre mémoire qui consiste en **le chapitre 3** permet d'exposer en détail la conception et l'implémentation: Nous avons commencé par une présentation de l'ensemble des outils utilisé, environnement et langage de programmation, modélisation de l'application, etc. Ensuite nous terminons par l'implémentation qui permet de définir et d'exploiter les différentes parties de l'application élaborée.

Et nous concluons par le rappel du bilan de notre travail et des résultats obtenus et quelques perspectives à donner à ce travail.

Chapitre 1: L'indexation

1. Problématique

Avec l'essor des techniques informatiques, la plupart des documents sont désormais produits et accessibles sous forme électronique. Différents dispositifs permettent de visualiser ensemble ou séparément tel ou tel élément, à tel niveau de granularité. Le défi consiste aujourd'hui à créer des outils informatiques performants pour consulter les documents, afin d'y rechercher et d'en extraire rapidement l'information pertinente. Les systèmes de recherche d'information se veulent ainsi comme des messagers dont l'objectif principal est de répondre aux interrogations des utilisateurs. Cependant, comme tout système bâti sur un autre, ils doivent leur efficacité au processus d'indexation qui s'efforce de donner une représentation fidèle, compacte et accessible de l'information recherchée. Pour mettre à la disposition de l'utilisateur l'information qu'il a du mal à trouver, il faut faire en sorte que les systèmes de recherche d'information soient plus efficaces et plus flexibles. Différentes moyen sont couramment utilisés pour chercher l'information dans un document. La recherche en texte intégral est coûteuse en temps d'accès. L'accès par la table des matières , qui est simple et direct, ne fournit que des titres comme point d'entrée : la table des matières sert surtout à donner une vue globale du contenu du document. La recherche à partir de l'index est plus souple et plus précise mais c'est aussi la plus coûteuse à concevoir.

Finalité de l'indexation :

L'indexation permet des rapprochements et des regroupements entre documents traitant des thématiques identiques ou liées. L'indexation vise à:

- décrire brièvement et clairement le contenu d'une collection de documents ;
- structurer le contenu thématique d'un ensemble de documents. Il permet ainsi d'effectuer des rapprochements et des regroupements entre documents traitant de thématiques identiques ou reliées ;
- améliorer la fiabilité et la qualité du repérage de l'information contenue dans les documents primaires.

2. Définitions

2.1. Indexation

L'indexation est une représentation qui permet de repérer et retrouver facilement l'information dans un ensemble de documents [Lancaster98]. C'est une activité de nature cognitive qui consiste à décrire le contenu d'un document a l'aide d'un langage d'indexation pour faciliter sa mémorisation dans un fichier, en vue d'une recherche ultérieure de l'information contenue dans ce document [Hudon98]. Elle consiste, après une analyse approfondie du contenu d'un document, à repérer, à sélectionner et à exprimer les informations contenues dans les documents primaires. L'indexation est une opération de description interne dont l'objet est le contenu intellectuel des documents. Les concepts sélectionnés pour représenter le contenu des documents primaires, sont exprimés au moyen de termes appartenant à un ou plusieurs documents. Elle peut également servir à comparer et classifier des documents, proposer des mots-clés, faire une synthèse automatique de documents, calculer des co-occurrences de termes, etc.

2.2. Index

Étant donnés les larges volumes de données à traiter, il est généralement admis que la recherche d'informations doit pouvoir s'appuyer sur une représentation synthétique des documents qui en résume les informations susceptibles d'être référencées par un utilisateur. Cette représentation est dénommée index. Un index est une structure qui permet d'associer à chaque terme d'indexation, la liste des documents qui contiennent ce terme. Il peut fournir d'autres informations que l'appartenance d'un terme à un document tels que le poids du terme dans le document et la position du terme dans le document (titre, auteur, titre d'un chapitre). [Abbaci03].

Un index d'après Hudon peut être une liste alphabétique ou systématique de termes choisis pour représenter les concepts et les sujets présents dans les documents constituant une collection, il fournit un renvoi clair à un document pertinent à une thématique, ou parfois même à l'endroit précis dans ce document où on trouvera un traitement du concept ou du sujet représenté [Hudon98].

2.3. Descripteurs

Les descripteurs représentent l'information atomique d'un index. Ils sont censés indiquer de quoi parle le document [Laporte00]. On parle aussi d'unités élémentaires (en anglais "tokens") [Jacquemin00]. Les descripteurs peuvent être choisi de tel manière a ce qu'on perde le moins

d'information sémantique possible, ils peuvent être [Pouliquen02]:

Des **mots** du document : toute chaîne de caractères compris entre deux séparateurs (espace, virgule, etc.). Exemple : "Les", "accidents", "vasculaires", "cérébraux".

Des **lemmes** : un processus supplémentaire appelé lemmatiseur convertit les documents pour extraire des mots de référence (ainsi les mots "CŒURS" et "CŒUR" aboutiront au même descripteur). La racinisation ("stemming"). Elle consiste à enlever des mots, les derniers caractères (considérés comme décrivant les flexions de mots). Exemple: enlever le "s" des mots au pluriel. Certaines racinisations utilisent des connaissances morphologiques plus complètes (suffixes, préfixes, etc.). Au lieu d'indexer un texte par des mots on l'indexe alors par le lemme correspondant. Exemple : "le", "accident", "vasculaire", "cérébral".

Des **concepts** : Il s'agit d'expressions (pouvant contenir un ou plusieurs mots). Ces concepts sont entrés manuellement (cas de l'indexation manuelle, ou semi-automatique). Ils doivent être choisis parmi une liste de concepts (on parle alors de vocabulaire contrôlé). Cette liste de concepts sera le plus souvent décrite dans un thésaurus (dans le cas des termes, on parlera de terminologie). Exemple des racines : "l", "accident", "vascul", "cérébr", ou concepts : "A.V.C.".

Des **N-grammes** : Il s'agit d'une représentation originale d'un texte en séquences de N caractères consécutifs. On trouve une utilisation de bigrammes et trigrammes dans la recherche documentaire (ils permettent de reconnaître des mots de manière approximative et ainsi de corriger des flexions de mots ou même des fautes de frappe ou d'orthographe). Ils sont aussi fréquemment utilisés dans la reconnaissance de la langue d'un texte[Harbeck99, Dunning94]. Exemple: "_l", "le", "es", "s_", "_a", "ac", "cc", "ci", "id", "de", "en", "nt", "ts", "s_", "_v", "va", "as" ... "au", "ux", "x_".

Des **contextes** : dans le cas de l'indexation sémantique latente (ou LSI, de l'anglais : Latent semantic indexation) [Deerwester90] les documents et leurs mots sont représentés sur d'autres dimensions. Cette indexation est le résultat d'une analyse des co-occurrences des mots dans un corpus (tout comme l'Analyse Factorielle des Correspondances). Le plus souvent, un système d'indexation avec vocabulaire contrôlé travaille à partir d'une base de connaissances. Celle ci, lui permet de "choisir" les descripteurs les plus appropriés en fonction du document analysé.

Le choix du type de descripteurs utilisé dans l'indexation est primordial, et sera déterminant pour les performances de l'indexation. La plupart des moteurs d'indexation fonctionnent sur les mots (notamment les moteurs de recherche sur le Web). La littérature anglophone fait le plus

souvent l'éloge de l'indexation par mots, une évaluation a même montré que la racinisation ("Stemming", la plus simple des méthodes linguistiques) n'améliorait pas de manière significative les performances [Pouliquen02].

3. Type d'indexations

L'indexation a pour un rôle de représenter le contenu des documents afin de permettre aux utilisateurs de les retrouver. Ces deux objectifs sont difficiles à réunir, en effet, la plupart des travaux montre que l'indexation est caractérisée principalement par la manière dont l'analyse du contenu et la traduction des concepts en langages d'indexation sont effectués, on distingue deux types : l'indexation par assignation et l'indexation par extraction [Hudon98].

3.1. Indexation par assignation (manuelle)

L'indexation par assignation (indexation humaine), se fait partir des concepts ou de l'idée qui se trouve derrière les représentations verbales et les mots. Hachani décompose l'indexation humaine en quatre étapes : la prise de connaissance du contenu du document, le choix des concepts, la traduction des concepts en descripteurs, l'établissement de liaisons syntaxiques entre les descripteurs. L'indexeur choisit ces mots clés dans une liste de vocabulaire contrôlé formé par le lexique ou le thésaurus, ce qui permet de garantir l'uniformité de la représentation du document. Mais le choix de ces outils de vocabulaire contrôlé dépendra de divers facteurs dont le type d'information recherchée, la finesse demandée à l'interrogation, la disponibilité et la compétence des indexeurs [Hachani97]. Le processus d'indexation humaine se déroule de la façon suivante : l'indexeur prend connaissance du document qui peut être une monographie, un article de périodique, des actes de communication d'un congrès. Il lit rapidement le titre, la table des matières, le résumé, l'introduction générale, les introductions et conclusions des principaux chapitres, l'intitulé des légendes et figures (s'il y en a) et la conclusion. Il parcourt en diagonale le contenu du document. Les différents modèles d'indexation manuelle sont les suivants :

1- *indexation dite « à plat »* où tous les descripteurs sont placés au même niveau d'importance par rapport au texte indexé.

2- *l'indexation pondérée* où on distingue les descripteurs principaux et secondaires, c'est la manière d'indiquer qu'un document traite prioritairement d'un sujet.

3- *l'indexation à rôles* où l'on souligne les relations qu'entretiennent entre eux les descripteurs

retenus pour mettre en évidence de manière plus fine le sujet du document.

Le processus d'indexation demande beaucoup de compréhension du domaine, de la langue, ainsi que le processus de transfert de l'information et l'utilisation des langages documentaires. L'indexation humaine est lourde à gérer, elle exige des analystes compétents, elle peut être performante à condition que le nombre de documents à indexer ne soit pas trop élevé.

3.1.1. Avantages de l'indexation manuelle

L'indexeur est un intermédiaire qui vient se glisser entre le document source et son utilisateur ultime. A l'étape de l'analyse, il doit juger de ce qui est important et de ce qui ne l'est pas. Il doit normaliser le nombre, la forme et la signification des termes d'indexation choisis. De ce fait, il en résulte un accès plus fiable, plus prévisible et généralement plus satisfaisant pour les utilisateurs qui veulent un repérage exhaustif.

3.1.2. Limites de l'indexation manuelle

L'analyse du contenu risque d'être biaisée par les connaissances préalables, les préjugés et les croyances de l'individu-indexeur, lesquelles peuvent s'opposer diamétralement à celles de l'utilisateur. L'indexation par assignation coûte cher et requiert du temps, elle peut mener à des problèmes de repérage puisqu'elle est souvent peu spécifique [Hudon98].

3.2. Indexation par extraction (automatique)

L'indexation par extraction est faite uniquement à partir des mots présents dans le document source. Elle est basée sur des méthodes de calcul statistiques ou statistico-linguistiques des mots et des phrases dans le texte numérisé [Hudon98].

3.2.1. Avantages de l'indexation automatique

L'indexation automatique d'un document disponible en format numérisé au moyen d'algorithmes plus ou moins complexes et sophistiqués présente de nombreux avantages. Elle permet d'offrir de longues «listes» de mots-clés très spécifiques sans délai et à coût réduit.

3.2.2. Limites de l'indexation automatique

Étant basée sur le contenu verbal d'un document source, l'indexation automatique ne peut indexer que les concepts, les thèmes ou les idées qui y sont représentés de façon explicite.

4. Langage d'indexation

Le terme d'indexation est généralement choisi au sein d'un langage d'indexation constitué au minimum d'un lexique et d'une syntaxe. Le lexique est formé de l'ensemble des termes utilisables pour la description, l'indexation et la recherche au sein d'un système particulier de transfert de l'information. La syntaxe du langage d'indexation est faite de l'ensemble des règles d'utilisation et de combinaison de ces termes [Hudon98]. Le langage d'indexation est un langage artificiel, il est construit à l'aide d'un ensemble de règles qui servent à la représentation abrégée du contenu d'un document [Sidhom02]. Le langage naturel, avec son lexique très étendu et ses règles syntaxiques complexes,peut lui-même servir de langage d'indexation; on parlera alors d'indexation en vocabulaire libre [Hudon98].

Indexation en vocabulaire libre	Indexation en vocabulaire contrôlé
Peu prévisible	Très prévisible
Plusieurs termes pour exprimer un même concept	Un seul terme pour exprimer un même concept
Stratégie de recherche complexe et coûteuse pour l'usager	Stratégie de recherche compacte et efficace
Bruit au repérage	Peu de bruit au repérage
Très haute spécificité	Plus général
Plus dynamique	Moins dynamique
Investissement minimal (formation des indexeurs, préparation d'outil, etc.)	Investissement important (formation des indexeurs, préparation d'outils, contrôle de qualité, etc.)
Grande flexibilité au niveau de la traduction des concepts	Limites imposées au niveau de la traduction des concepts

Tableau 1: Les avantages et inconvénients de l'indexationen vocabulaire libre et en vocabulaire contrôlé [Lévesque02].

5. Méthodes d'indexation automatique

De nombreuses méthodes (linguistiques, statistiques, etc.) ont été développées pour concevoir, ou améliorer les systèmes et les logiciels d'indexation automatique.

5.1. La méthode linguistique

En indexation, le texte et les descripteurs utilisés pour la représentation du document font appel

à la linguistique. De plus, le fait que certains systèmes d'indexation utilisent les techniques du traitement automatique des langues, démontre la pertinence de l'approche linguistique. L'absence de syntaxe lors des interrogations peut provoquer un bruit important au niveau du résultat. C'est pourquoi des méthodes basées sur une désambiguïsation syntaxique ont été élaborées. Ces méthodes vont affecter à chaque descripteur (un rôle). On peut citer deux systèmes qui fonctionnent sur ce modèle : PRECIS de D. AUSTIN, adopté par la Bibliographie Nationale Britannique (BNB) et SYNTOL, réalisé en 1960 par J.C. GARDIN et ses collègues du CNRS et de la Maison des sciences de l'homme avec le soutien de l'Euratom [Hachani97].

Ces travaux permirent d'ouvrir la voie à divers autres travaux qui aboutirent à la conception de logiciels, expérimentés pour la plupart sur des banques de données de différents organismes importants. On peut citer PIAF-DOC expérimenté par la Documentation Française ou encore le système SINTEX de la Société d'Information Européenne (SIE), expérimenté sur les banques de données des Communautés Européennes, ECODOC et CELEX. Ces systèmes basés sur une analyse syntaxique ne sont pas très performants lors de l'interrogation. Le taux de bruit et de silence ont certes été réduits par rapport aux systèmes de départ et l'analyse syntaxique a permis de nombreuses avancées. Des travaux, axés sur l'analyse de contenu et sur l'analyse de textes ont permis de concevoir d'autres systèmes dont notamment les analyseurs syntaxiques et morphologiques qui sont apparus dans la décennie 1980. On peut citer les logiciels AlexDoc[1] de la société GSI-Erli et SPIRIT de la société SYSTEX qui permettent une indexation sur texte intégral et sur résumé ainsi que l'interrogation des bases de données en texte intégral par le biais d'une indexation des questions.

Les systèmes combinant différentes analyses linguistiques pour le traitement en langage naturel sont formés de plusieurs modules de traitement linguistique ayant chacun une analyse spécifique :

- niveau morphologique : Chaque terme est isolé par le biais d'un dictionnaire qui permet le contrôle des chaînes de caractères et le repérage des mots,
- niveau lexical : Le traitement se traduit par la suppression des variantes combinatoires (flexion, dérivation, conjugaison) pour obtenir une forme canonique par réduction ou lemmatisation. Les outils nécessaires à ce procédé de réduction sont des dictionnaires de correspondances entre formes fléchies ou dérivées et formes canoniques (produira,

1 AlexDoc est l'une des couches d'applications du logiciels d'enregistrement et de recherche documentaire ALEXIS. Il comporte des programmes d'aide à l'indexation des documents et des questions.

produisent, ont produit etc., auront la même forme canonique produire) ainsi que des règles d'établissement par correspondance.

Les traitements morphologique ou lexical engendrent des ambiguïtés sémantiques qui peuvent être levées par d'autres analyses linguistiques :

- la syntaxe permet de résoudre des ambiguïtés mais elle est impuissante face à certaines ambiguïtés de la langue naturelle.
- le niveau pragmatique, l'être humain fait appel à ses connaissances du monde et du contexte pour résoudre les cas de polysémie. Dans le cas du traitement automatique, on fait appel aux réseaux sémantiques, mais ces réseaux ne concernent que des domaines très spécialisés et sont d'utilisation restreinte.

D'autres recherches en traitement linguistique de l'information ont été menées pour améliorer la qualité de l'indexation automatique. On s'est intéressé à la sélection des mots isolés qui représente la faiblesse majeure de l'indexation automatique. En effet la reconnaissance des syntagmes nominaux (la plus petite unité porteuse de sens dans une phrase) et de leur fonction syntaxique permettraient une recherche plus fine. Des analyseurs morpho-syntaxiques ont été élaborés dans ce sens. Plusieurs équipes de recherche ont travaillé sur cet aspect du syntagme nominal, on peut citer le groupe SYDO à Lyon, ou d'autres équipes du monde entier, toutes les langues ou presque ont en un (portugais, espagnol, japonais, arabe, etc.). Le problème de ce type d'analyse est l'ambiguïté de du langage naturel. Michel LE GUERN du groupe SYDO préconise de prendre en compte les deux solutions pour éviter que lors de l'interrogation, l'utilisateur ne soit en face d'un silence du système [Hachani97].

5.2. La méthode statistique

H. P. Luhn déclare qu'au lieu de tirer l'information au hasard comme le fait normalement le lecteur, la nouvelle méthode automatique choisit les phrases d'un article qui représentent le mieux l'information pertinente [Luhn58]. Il ouvrit la voie aux travaux sur l'indexation automatique par voisinage appelée aussi méthode statistique. La fréquence d'un mot dans un article fournit la mesure utile de la signifiance d'un mot. Plus certains mots sont souvent rencontrés en compagnies les uns des autres dans une phrase plus on peut dire que ces mots sont lourds de sens [Hachani97].

Les méthodes statistiques, sont basés sur deux types de traitement : le premier est basé sur le

calcul de fréquence statistique (avec prise en compte des synonymes), de fréquence selon une table (table spécialisée par corpus homogène ou table selon la loi de Zipft). Le deuxième type de traitement est construit sur une recherche de voisinage (que l'on appelle également méthodes par co-occurrence) : avec ou sans élimination de polysémies ou avec le calcul de la distance moyenne [Hachani97]. La méthode statistique est basée sur le mot plein. En effet, plus un mot plein est présent dans un texte, plus il est signifiant et servira de descripteur [Hachani97].

5.3. La méthode mixte

Aujourd'hui de nombreux outils, logiciels ou systèmes sont basés non pas sur une méthode d'indexation mais combinent plusieurs analyses en même temps. Il s'agit pour la plupart d'une combinaison linguistique (morphologie, syntaxe, lexique, sémantique) et statistique. On peut citer: SPIRIT de T-GID, STAND d'IBM France. Ces systèmes permettent une indexation en langage naturel et intègrent des outils d'analyse linguistique multilingue [Hachani97]. Ainsi SPIRIT, et MICRO-MIND, développé par la suite par la même société, représentent un système de documentation automatique dont toutes les procédures de l'indexation à l'interrogation sont entièrement automatisées, l'interrogation se fait en langage naturel [Hachani97]. Ils sont composé de quatre modules :

- un module linguistique (dictionnaire morphologique, analyseur syntaxique),
- un module fichier inverse qui permet d'accéder aux documents à partir de concepts pondérés et ce dans un temps optimisé,
- un module statistique (ou probabiliste) qui permet d'affecter à chaque concept une fonction de poids informationnel et d'ordonner les réponses en fonction de leur pertinence,
- un quatrième et dernier module d'interrogation en langage naturel et de proximité sémantique.

SPIRIT intègre des dictionnaires généraux (français, anglais, allemand), pour l'interrogation des bases de données multilingues.

6. Règles de l'indexation

L'indexation obéit à un certain nombre de règles qui peuvent être réparties en trois groupes :

6.1. Les règles de sens

La première étape dans l'élaboration d'un index consiste en une recherche du vocabulaire pertinent dans le domaine traité, en vue d'une sélection des termes et expressions retenus pour l'indexation, parmi les règles de sens, on cite:

6.1.1. La règle de pertinence

L'indexeur doit veiller, au moment de l'identification et de la sélection des concepts à représenter dans l'index, à ne choisir que des termes adaptés à la fois à l'expression des notions contenues dans le document primaire et à l'usage normal du vocabulaire. Cet usage normal peut, selon les cas, s'étendre de celui du grand public à celui des spécialistes d'un domaine. On le détermine en analysant le niveau de spécialisation des documents. Une bonne politique d'indexation permet déjà de régler cette question. En outre, le type d'indexation adoptée joue un rôle important. L'indexation par extraction permet d'assurer une correspondance entre les notions repérées dans le document et les termes d'indexation.

6.1.2. La règle de non-inclusion

Il faut indexer un document sous un sujet spécifique et non sous une classe qui englobe ce sujet. L'indexation alphabétique par matières est un langage documentaire non hiérarchisé. Cette structure permet de donner un accès direct à un grand nombre de notions générales ou particulières. Ainsi, un ouvrage sur les chats sera indexé par chat et non par Zoologie.

6.1.3. La règle d'efficacité

Elle signifie que tous les termes d'indexation choisis sont utiles et que chaque notion n'est exprimée que par un seul terme qui ne sert à exprimer qu'une seule notion. En effet, le langage documentaire vise à éliminer les ambiguïtés, les mots vides, les synonymes, la polysémie du langage naturel.

6.1.4. La règle d'objectivité

L'indexeur doit adopter une attitude de neutralité face au contenu des documents analysés. Les notions choisies pour décrire le contenu doivent être exprimées dans un souci de neutralité. Ainsi des termes subjectifs ou relatifs par rapport au temps, tels que les mots "essor", "déclin", "décadence", qui manifestent une certaine vision d'un processus historique dont la subjectivité n'est pas absente doivent être remplacés par des indications objectives de dates. Il en est de même des

adjectifs comme ancien, nouveau, moderne, contemporain.

6.2. Les règles de forme

On retient généralement quatre règles de forme :

6.2.1. La règle de l'emploi du français

L'emploi des termes français doit être préféré à celui de formes étrangères, si les utilisateurs sont francophones. Cependant, certains termes étrangers d'emploi courant en français qui désignent des notions spécifiques à une culture étrangère, sans équivalents exacts en français sont admis comme tels. Exemple : Marketing

6.2.2. La règle de l'emploi du singulier

Cette règle vise à faire apparaître sous une forme unique les termes d'indexation. Elle doit être respectée d'une façon générale ; cependant elle connaît certaines exceptions. Nous citons, l'exemple des mots n'existent qu'au pluriel ou sont employés de préférence sous cette forme.

Exemples : Archives, Finances publiques, Relations internationales, Mathématiques

6.2.3. La règle de l'emploi de la forme développée des sigles

Cette règle vise à éviter les confusions ou les difficultés de compréhension provoquées par l'emploi d'un grand nombre de sigles. L'exception peut être faite pour les sigles très connus comme UNESCO.

6.2.4. La règle de l'emploi des majuscules

Le premier mot d'un terme d'indexation, ainsi que les noms propres commencent par une majuscule. On suit les règles de l'orthographe.

6.3. Les règles de structure

Elles dépendent de la syntaxe du langage documentaire adopté. Les expressions sont formées d'un mot seul ou d'une association de mots formant une expression, les mots sont écrits en minuscules, les sigles en majuscule et sans point entre les lettres.

7. Ressources linguistiques

Les différentes méthodes d'indexation automatique utilisent des ressources linguistiques. Les plus couramment utilisées sont les lexiques, les thésaurus et les corpus de textes.

7.1. Lexiques

Un lexique est un ensemble de mots répertoriés qui est utilisé dans la phase d'indexation. Celui-ci peut contenir toutes les flexions des mots afin de reconnaître un mot quelle que soit son orthographe. Un lexique contient au minimum l'orthographe de chaque mot. On lui associe souvent des relations de flexions, de synonymies, de dérivations. Chaque entrée du lexique peut avoir une étiquette grammaticale (mot singulier, pluriel, nom propre, masculin, féminin, adverbe). Il peut également contenir une liste de mots composés, des relations sémantiques entre les mots. Si le lexique contient la définition de chaque mot, on parle alors de dictionnaire.

Il existe des lexiques informatiques que peuvent utiliser les outils linguistiques (exemple: Wordnet et Eurowordnet [Pouliquen02]).

7.2. Thésaurus

Un thésaurus peut être défini comme un répertoire de termes normalisés. Le plus souvent le thésaurus contient aussi un réseau sémantique reliant ces termes (synonymes, termes reliés, termes spécifiques, termes génériques). Felber définit un "terme" comme « représentant linguistique d'un concept dans un domaine de connaissance » [Pouliquen02]. Jacquemin le définit comme le « résultat d'un processus d'analyse terminologique », qui permet de définir un mot ou une unité lexicale comme terme par décision humaine [Jacquemin00].

Il existe des outils de constitution de thésaurus. Ils se basent sur un corpus de textes existant et par des méthodes statistiques, linguistiques ou mixtes. Parmi les logiciels les plus importants d'acquisition de ressources terminologiques citons: Termino, Ana, Acabit, Lexter, Xtract [Pouliquen02].

7.3. Corpus de textes

L'indexation se fait en général sur un "ensemble" de textes, appelé corpus de documents. Ce corpus peut être local (ensemble de rapports, de courriers), ou extérieur (CDROM, documents téléchargés sur Internet). Plus le corpus de texte est homogène, plus le vocabulaire sera homogène, et plus le processus d'indexation sera performant.

8. Conclusion

L'indexation est née du besoin grandissant de recherche d'informations, c'est une pratique référencée aux utilisateurs qui utilise des techniques destinées à représenter et fabriquer des outils de recherche, d'accès et de navigation dans les documents au sein de large corpus. L'indexation substitue au document une notice documentaire, à partir de laquelle seront effectuées les recherches.

Dans ce chapitre nous avons cité une panoplie de méthodes d'indexation manuelle et automatique et nous avons souligné l'évolution qu'a impliquée l'entrée de l'informatique dans le domaine de la documentation qui a impliqué un gain de productivité.

L'indexation a permis par rapport au volume d'information à traiter souvent croissant, une facilité et une rapidité à réindexer une base entière après mise à jour du langage documentaire sur lequel l'indexation automatique s'appuie et une meilleure qualité à l'inverse de l'indexation humaine, qui est moins stable au sens où un même document n'est pas indexé de la même manière par deux personnes à deux moments différents. De plus, l'intérêt pour l'indexation automatique grandit lorsque le renouvellement des indexeurs humains est important. Elle permet ainsi de réduire les coûts de l'indexation.

Chapitre 2: L'indexation automatique

1. Introduction

Aujourd'hui la plupart des systèmes tendent vers une combinaison de différentes méthodes d'indexation et non pas sur une seule, ils gagnent à être plus performants. L'approche d'indexation mixte propose d'utiliser les meilleurs méthodes décrits précédemment afin de les combiner. Le processus d'indexation mixte permet d'associer à chaque document un descripteur (index), le plus souvent sous forme d'un ensemble de mots (termes d'indexation), en utilisant un processus automatisé. Avant de transformer ces mots en forme d'origine, ils sont filtrés en comparant avec les termes qui n'ont pas de valeur d'indexation (stop-list). En compte aussi le poids de chaque terme à partir de la fréquence du terme. Finalement, les termes, les poids, les identités des documents sont enregistrés dans la base d'index.

2. Recherche d'information

La Recherche d'Information (RI) est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche et la sélection d'information [Tebri04].

Selon , la RI propose de retrouver parmi une masse importante de documents, ceux qui peuvent répondre aux besoins d'un utilisateur exprimés, en général, par des requêtes en langage naturel [Zargayouna05].

Un système de recherche d'information (SRI) est un outil informatique qui permet à l'utilisateur d'exprimer son besoin d'information à l'aide d'une requête et qui retrouve les documents pertinents à cette requête parmi l'ensemble des documents qu'il gère, ensemble appelé corpus du système [Abbaci03].

3. Modèles de Recherche d'information

Le modèle joue un rôle central dans la Recherche d'Information dans la mesure que c'est lui qui détermine le comportement clé d'un SRI. Il doit fournir une formalisation du processus d'indexation et accomplir plusieurs rôles dont le plus important est de fournir un cadre théorique pour la modélisation de la mesure de pertinence [Sauvagnat05].

Le modèle doit d'abord donner une signification au résultat de l'indexation, un document est représenté par un ensemble de termes index et leurs poids, si un terme index est sensé représenter un concept important décrit dans un document, il existe différentes façons d'interpréter son poids,

un modèle théorique doit donner une interprétation précise à ce poids.

Le modèle doit aussi interpréter les relations possibles entre les termes d'indexation, ces deux fonctions nous amènent à la représentation d'un document, une représentation similaire peut être créée pour une requête. Finalement, un modèle de recherche d'information doit déterminer la relation entre un document et une requête à partir de leurs représentations. Ceci se fait souvent avec un calcul de similarité [Dahak06].

4. Taxonomie des modèles de RI

Les modèles de RI peuvent être classés, suivant le cadre théorique sur lequel ils s'appuient, en trois principaux [Sauvagnat05, Moreau06]:

- **Les modèles ensemblistes**, dont le représentant le plus connu est le modèle booléen. Dans ce type de modèles, les opérateurs logiques (*OR*, *AND*, *NOT*) séparent les termes de la requête et permettent d'effectuer des opérations d'union, d'intersection et de différence entre les ensembles de résultats associés à chaque terme.
- **Les modèles algébriques**, dont le premier représentant a été le modèle vectoriel. Dans ces modèles, la pertinence d'un document vis-à-vis d'une requête est définie par des mesures de distance dans un espace vectoriel.
- **Les modèles probabilistes**, reposent sur la théorie de probabilités. Dans ces modèles, la pertinence d'un document vis-à-vis d'une requête est vue comme une probabilité de pertinence document/requête.

La **Figure 1** illustre les principaux modèles de RI existant.

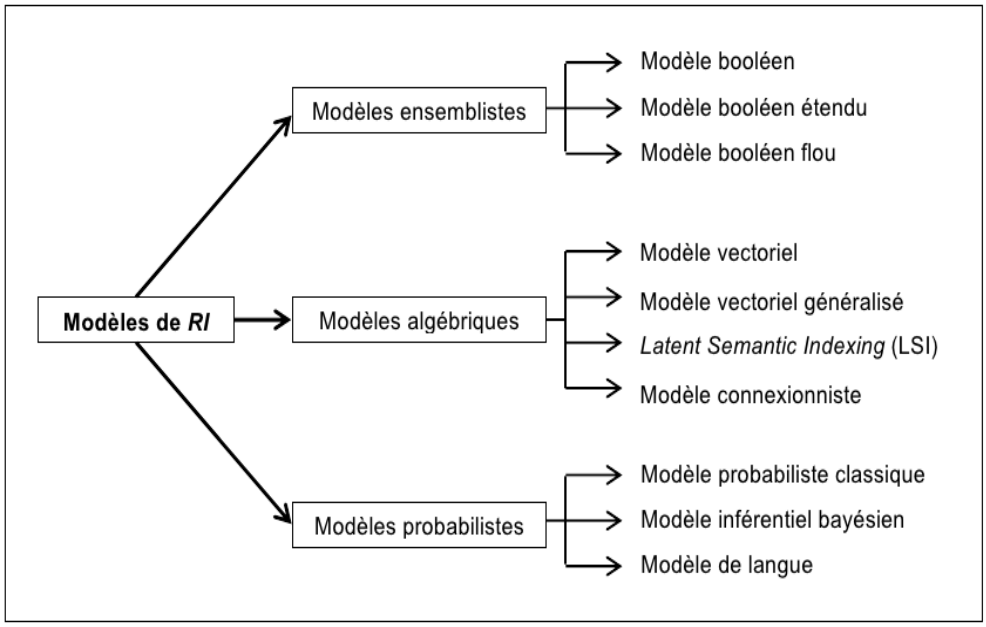

Figure 1: Taxonomie des modèles de RI.

Pour chaque catégorie on va citer le modèle le plus connu.

4.1. Le modèle booléen

Le modèle booléen tire son nom des opérateurs booléens utilisés pour formuler une requête. En effet, une requête est une formule logique, combinant des descripteurs et des opérateurs (et, ou, non). Les documents sont représentés par une liste de descripteurs. Ces descripteurs peuvent appartenir à un langage libre ou contrôlé. Ils peuvent être extraits automatiquement des documents ou choisis par des documentalistes. Les index sont stockés dans un fichier inverse où, à chaque descripteur correspond la liste des documents contenant ce descripteur dans leur index. La fonction de comparaison retrouve les documents dont les index valident la formule logique de la requête. La base de documents est séparée en deux, les documents qui correspondent à la requête et ceux qui ne correspondent pas [Roussey01].

Le principal avantage du modèle booléen est sa transparence, un document n'est sélectionné que dans le cas où il répond exactement au besoin de l'utilisateur. Cependant, ce modèle a également des inconvénients tels que la difficulté d'exprimer des requêtes complexes pour un utilisateur simple. De plus, l'absence d'un seul terme de la requête dans un document empêche la sélection de ce dernier. Aussi il ne permet pas de classer les documents retournés.

Il existe deux autres modèles basés sur les ensembles: le modèle booléen étendu et le modèle

booléen flou [Bensiali07].

4.2. Le modèle vectoriel

Le modèle vectoriel représente un document ou une requête par un vecteur dans un espace de termes c'est-à-dire l'espace d'indexation construit à partir des entités d'indexation. Les coordonnées des vecteurs sont les poids indiquant l'importance du descripteur par rapport au document. L'ensemble des coordonnées des vecteurs est contenu dans une matrice. La fonction de comparaison évalue la correspondance entre deux vecteurs (document et requête) ce qui permet de classer les résultats. Le schéma de la **Figure 2** illustre cette méthode [Roussey01] :

T_k est un des vecteurs de base de l'espace de représentation. Il représente l'entité d'indexation k.

D_i est le vecteur désignant le document i.

$W_{i,k}$ est le poids de l'entité k dans le document i.

$D_i = (W_{i,1}, W_{i,2}, W_{i,3})$.

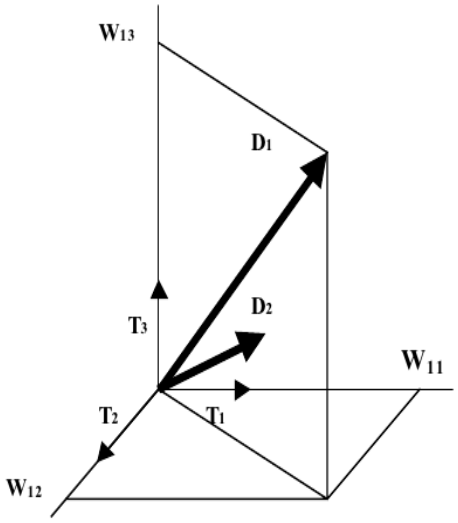

Figure 2: Représentation des documents dans un espace vectoriel intitulé espace des termes.

L'inconvénient majeur du modèle vectoriel est le fait qu'il considère que les termes de l'index (la représentation du document) sont tous indépendants [Sauvagnat05]. Malgré sa simplicité, le modèle vectoriel reste supérieur ou au moins aussi bon que les autres modèles qui tiennent compte des dépendances entre les termes, et c'est pour toutes ces raisons qu'aujourd'hui le modèle vectoriel

est le plus populaire en recherche d'information.

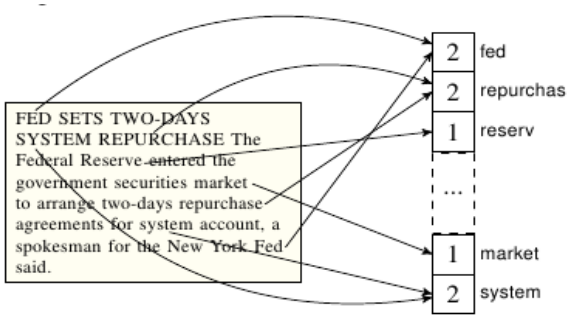

Figure 3: Représentation vectorielle d'un texte.

4.3. Le modèle probabiliste

Ce est modèle basé sur la théorie des probabilités, elle considère la recherche d'information comme un espace d'événements possibles. Un événement peut être le jugement de pertinence porté par l'utilisateur sur un document par rapport à une requête ou l'association d'un descripteur à document.

La représentation des documents est généralement un index pondéré, les poids des descripteurs correspondent à la probabilité que le descripteur soit pertinent pour le document, elle est aussi appelée degré de croyance [Roussey01].

Selon [Piwowarski03], toutes les approches probabilistes partent du même principe, ou plutôt de la même question: prenons un document *d* et une question *q*, quelle est la probabilité que ce document réponde à cette question ? De manière formelle, cette probabilité est notée $P(R|q,d)$. Il s'agit de la probabilité d'observer une relation de pertinence *R*, sachant qu'on observe le document *d* et la question *q*.

Le modèle probabiliste le plus connu est le modèle basé sur les réseaux bayésiens [Bensiali07].

5. Processus d'indexation

L'indexation consiste à analyser chaque document afin d'en extraire un ensemble de mots susceptibles de représenter au mieux son contenu. Ces derniers seront exploités par le SRI lors du processus de recherche.

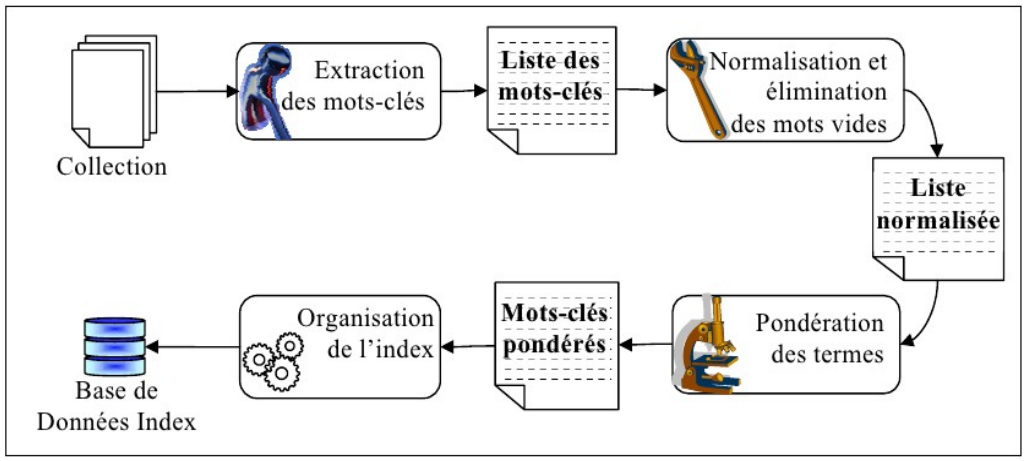

Figure 4: Étapes du Processus d'indexation [Dahak06]

L'indexation consiste d'abord à extraire les termes du document n'apparaissant pas dans un anti-dictionnaire (stop-list[2]) et dans une certaine limite de fréquence. Les termes sélectionnés sont ensuite lemmatisés, ce qui permet une certaine normalisation des termes dans l'index [Paradis96].

5.1. Segmentation des mots (Tokenization)

La tokenisation est l'opération de segmenter un acte langagier en unités "atomiques" (tokens). Les tokenisations les plus courantes sont le découpage en mots ou bien en phrases. Ainsi la tokenisation en mots de la phrase « Tout le monde attendait une véritable furia rouge. » nous donnerait les composants mots : « Tout », « le », « monde », « attendait », « une », « véritable », « furia », « rouge », « . ».

En traitement automatique des langues, la tokenisation est surtout utilisée lors des prétraitements afin d'identifier des unités de "haut niveau" sur lesquelles porteront l'analyse. De plus, le terme "tokenisation" est couramment associé à l'idée de "découpage en mots", soit une analyse morphologique des actes langagiers.

C'est un traitement de surface assez simple dans le principe, mais particulièrement difficile à réaliser de manière exacte, les documents ayant beaucoup de bruits et des représentations assez variées. Notons que des méthodes de segmentation moins intuitives ont été proposées pour des problèmes spécifiques comme par exemple la segmentation en *n-grams* de lettres pour le traitement de corpus multilingues [Denoyer04].

2 stop-list : Liste des mots fréquemment utilisés dans une langue donnée. Donc insignifiants en terme de description du contenu d'un document. Exemple : la, le, de, dans, en, je...

5.2. Sélection des mots d'indexation

Dans un document, tous les mots n'ont pas la même signification. Si le mot est rare, la signification sur une occurrence de mot devient plus grande. Un mot s'appelle « mots-vides » s'il contient peu d'information. Par exemples : les mots grammaticaux : prépositions (à, de), articles (le, la, un, des), auxiliaires (être, avoir), pronoms... Il faut éliminer ces mots en utilisant un dictionnaire, une liste (stop-list) recensant tous les mots non pertinents pour l'indexation. Un mot est accepté seulement s'il ne fait pas partie de stop-list.

Certains mots inclus dans cette liste ne sont pas nécessairement vides de sens (ça dépend du domaine. Ils ne sont pas vides de sens en linguistiques). Mais leur sens importe très peu pour des besoins de RI.

La liste utilisée dans un système peut aussi varier. Cela dépend du domaine d'application. Par exemple, le mot "article" est inclus dans certains systèmes comme mot vide parce qu'on reçoit beaucoup de requête d'utilisateur qui contient le mot "papier", comme "des papier sur l'informatique". Cependant, ce mot peut être très significatif dans certaines applications (par exemple, pour une base de documents en papeterie) [Dahak06].

5.3. Les lemmes et stemmes

La lemmatisation consiste à représenter chaque mot par sa forme canonique. Ainsi, les verbes par leur forme infinitive et les noms par leur forme au singulier sont pris en compte. Le processus permet une réduction du nombre de descripteurs. Par exemple, le remplacement des mots conductrice, conducteur par l'unique racine conducteur semble être avantageux tout comme le remplacement des formes conjuguées franchit et franchi par le lemme franchir [Laroum09].

La stemmatisation (radicalisation) consiste à supprimer tous les affixes d'un mot. Par affixes on entend : suffixe (défin-**ition**), préfixe (**sur**-consommation), le stemmer le plus utilisé est le stemmer de Porter [Denoyer04].

Par exemple, pour la phrase « le chat est le chat de ma voisine » :

*Le Stemming: « l*e, cha, es, le, cha, de, ma, voisin »

Les mots « chat » et « voisine » ont été réduits à leur racine.

La Lemmatisation : « le, chat, être, le, chat, de, mon, voisin »

Les mots ont été remplacés par leur forme générique (par exemple, « est » a été remplacé par

l'infinitif « être »).

Ces techniques sont principalement utilisées pour réduire l'espace de représentation des documents et faire ressortir les traits similaires entre les mots. Notons que l'utilisation des n-grams présentés précédemment [1.2.1] est une forme de lemmatisation dans le sens où elle permet de conserver les racines des mots. Les stemmer sont plus performants mais spécifiques à une langue tandis que les n-grams sont moins performants mais peuvent regrouper des mots de différentes langues issus de la même racine.

5.4. La pondération des mots-clés

La pondération est entièrement dépendante du modèle de recherche d'information utilisé. Elle permet de définir l'importance qu'a un terme dans un document donné, elle est également utilisée pour filtrer l'index résultant du processus d'indexation (c'est-à-dire : éliminer les index dont le poids est inférieur à un certain seuil). Il existe plusieurs techniques de pondération des termes, nous citons les plus importantes [Dahak06].

5.4.1. La fréquence d'occurrences

Un mot qui apparaît souvent dans un texte représente un concept important (Loi de Zipf). La représentation fréquentielle est une extension naturelle de la représentation binaire (Attribution d'un poids 1 pour l'existence et 0 pour l'absence) qui prend en compte le nombre d'apparitions d'un mot dans un document. Ainsi, un document est représenté dans l'espace V et chaque composante correspond au nombre d'apparition du terme dans le document [Denoyer04]. De manière formelle, si d_{freq} est la représentation fréquentielle du document d de composantes d^{i}_{freq} pour i ∈ [1.. V], nous pouvons écrire :

$$\forall i \in [1..|V|], d^{i}_{freq} = \textit{nombre d'apparitions du terme dans } d$$

Formule 1: Fréquence d'occurrence.

La loi de Zipf dit que les termes les plus informatifs d'un corpus de documents ne sont pas les mots qui apparaissent le plus dans le corpus car ceux-ci sont pour la plupart des mots outils (du type article, mots de liaison etc....) ni les mots les moins fréquents du corpus qui, quant à eux peuvent par exemple être issus de fautes d'orthographe ou de l'utilisation d'un vocabulaire trop spécifique à quelques documents du corpus. Par contre, un mot qui apparaît beaucoup dans un document

possède certainement une information forte sur la sémantique du document. Les deux considérations précédentes ne sont pas opposées et peuvent être résumées ainsi, de manière peu formelle : «Un mot est informatif dans un document si il y est présent souvent mais qu'il n'est pas présent trop souvent dans les autres documents du corpus» [Denoyer04]. Cette loi illustre donc, l'importance d'un terme en fonction de sa fréquence dans un corpus. Un mot est important si il n'est ni trop fréquent ni trop rare. Il devient évident que ce ne sont pas tous les mots les plus fréquents qui sont des index représentatifs. On peut donc, définir un seuil maximal: si la fréquence d'un mot dépasse ce seuil, alors il n'est pas considéré comme représentatif. L'utilisation d'un seuil minimal et maximal correspond à l'informativité d'un mot, qui mesure la quantité du sens qu'un mot porte. Cette notion n'est pas définie très précisément dans la RI. Elle est utilisée seulement de façon intuitive. La correspondance entre l'informativité et la fréquence d'apparition des mots est illustrée dans la Figure suivante:

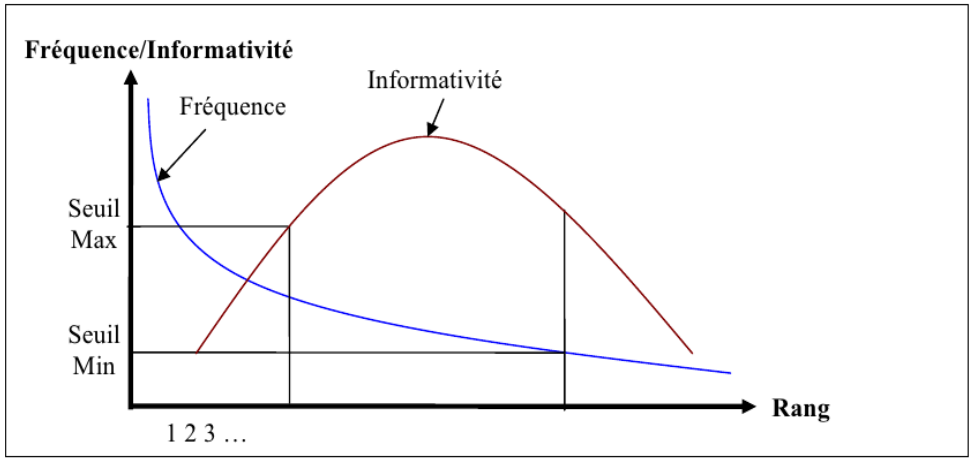

Figure 5: La correspondance entre l'informativité et la fréquence [Rijsbergen80]

Le rang représente le mot le plus fréquent d'un texte (rang=1).

Ainsi, les mots dont les fréquences sont entre les deux seuils ont une informativité plus élevée. Ils sont donc les plus représentatifs de l'information.

Note : L'inconvénient majeur de la représentation fréquentielle vient du fait qu'un document de longueur élevée sera représenté par un vecteur dont la norme sera supérieure à celle de la représentation d'un document plus court. Il est donc plus habituel de travailler avec une version normalisée de la représentation fréquentielle où chaque composante du vecteur de représentation

code la proportion d'un terme dans le document.

5.4.2. La valeur de discrimination

Par "discrimination", on réfère au fait qu'un terme distingue bien un document des autres documents. C'est-à-dire, un terme qui a une valeur de discrimination élevée doit apparaître seulement pour un petit nombre de documents. Un terme qui apparaît dans tous les documents n'est pas discriminant. Le pouvoir de discrimination d'un terme est important dans le choix des index qu'on veut garder [Rijsbergen80].

Pour calculer la valeur de discrimination, chaque document est représenté par un vecteur de poids p_{ij} (le poids du terme t_j dans le document d_i). Étant donné un corpus, on a donc une matrice. Pour calculer la valeur de discrimination d'un terme, on doit comparer une sorte d'uniformité au sein du corpus avec celle du corpus transformé dans lequel le terme en question a été uniformisé (mis au même poids). L'idée est que, si on uniformisant le poids d'un terme dans tous les documents, on obtient une grande amélioration dans l'uniformité du corpus alors ce terme est très différent dans différents documents : Il a donc une grande valeur de discrimination. En revanche, si en uniformisant le poids du terme, on n'obtient pas beaucoup d'amélioration sur l'uniformité, ce terme est déjà distribué de façon uniforme, donc peu discriminant.

Le calcul de la valeur de discrimination d'un terme se fait comme suit:

1) On calcule d'abord le vecteur centroïde (ou le vecteur moyen) du corpus: Pour chaque terme, son poids dans le vecteur centroïde V est le poids moyen de ses poids dans les documents. C'est-à-dire, si N est le nombre de documents dans le corpus on aura:

$$p_{ij} = \sum_i p_{ij} / N$$

Formule 3: Vecteur Centroide.

2) On calcule l'uniformité du corpus comme la similarité moyenne des documents avec le vecteur centroïde:

$$U_1 = C * \sum_j RSV(d_i, V)$$

Formule 4: Uniformité de corpus.

Où ***C*** est une constante de normalisation (par exemple ***1/N***), et ***RSV(d_i,V)*** est la similarité entre le document ***d_i*** et le vecteur centroïde ***V*** (telle que définie pour le modèle vectoriel).

3) On uniformise le poids du terme en question à ***0***, et on répète les deux étapes ci-dessus pour obtenir une nouvelle valeur d'uniformité ***U_2***.

4) La valeur de discrimination du terme est:

$$V = U_2 - U_1$$

Formule 5: La valeur de discrimination

Note : Dans ce calcul de la discrimination, on ne se préoccupe pas beaucoup de la fréquence d'un terme dans un document particulier, mais beaucoup plus de sa distribution dans le corpus.

*5.4.3. Tf*Idf*

Tf*idf désigne un ensemble de schémas de pondération de termes. Par ***tf***, on désigne une mesure qui a un rapport avec l'importance d'un terme pour un document. En général, cette valeur est déterminée par la fréquence du terme dans le document. Par ***idf***, on mesure si le terme est discriminant (ou non-uniformément distribué) [Salton88]. On présente ci-dessous quelques formules les plus utilisées de ***tf*** et ***idf***.

$$tf = f(t,d) / Max[f(t,d)]$$
$$tf = \log(f(t,d))$$
$$tf = \log(f(t,d)+1)$$

Formule 6: Term frequency (tf)

Où ***f(t,d)*** est la fréquence d'occurrence du terme ***t*** dans le document ***d***;

$$idf = \log(N/n)$$

Formule 7: Inverted Document Frequency (Idf)

Où ***N*** est le nombre de documents dans le corpus, et ***n*** ceux qui contiennent le terme ***t***.

Une formule de **tf*idf** est donc la multiplication d'une ***tf*** par une ***idf***. Par exemple:

$$tf * idf = [f(t,d) / Max[f(t,d)]] * \log(N/n)$$

Formule 8: tf*idf

Une formule **tf*idf** combine les deux critères qu'on a vu: l'importance du terme pour un document (par ***tf***), et le pouvoir de discrimination de ce terme (par ***idf***). Ainsi, un terme qui a une valeur de **tf*idf** élevée doit être à la fois important dans ce document, et aussi il doit apparaître peu dans les autres documents. C'est le cas où un terme correspond à une caractéristique importante et unique d'un document.

Tous comme pour le cas des représentations fréquentielles, le vecteur **TF-IDF** sera habituellement normalisé afin d'éviter les problèmes posés par les différentes longueurs de documents. On parlera alors de représentation **TF-IDF** normalisée [Denoyer04].

5.5. Indexation Sémantique Latente

Une technique conçus pour résoudre les problèmes de synonymie et polysémie, elle permet d'observer le contexte dans lequel apparaissent les mots clés au sein des documents.

Le principe général de l'analyse de la sémantique latente [Landauer97] consiste à définir la signification des mots à partir des contextes dans lesquels ils apparaissent au sein de vastes corpus de textes. Il est en effet possible de déterminer le sens d'un mot à partir de son contexte, dès lors que ce mot est rencontré suffisamment souvent. LSA analyse les contextes d'occurrence des mots au sein d'un vaste corpus et réduit le bruit causé par la variabilité de l'emploi de ces mots dans la langue.

Chaque mot est représenté par un vecteur dans un espace de plusieurs centaines de dimensions, c'est-à-dire par une suite de centaines de valeurs numériques. Cette représentation vectorielle permet aisément de calculer un vecteur pour une suite de mots, voire un texte entier, en ajoutant simplement les vecteurs des mots qui les composent. Ce formalisme de représentation ne possède pas le côté explicite des représentations symboliques du sens, mais il compense cela par une métrique objective rendant possible des comparaisons de signification entre mots ou groupe de mots, de manière complètement automatique. Par exemple, les deux phrases « J'ai perdu mon chat dans la forêt. » et « Le petit félin a disparu dans les arbres. » sont représentées par deux vecteurs qui sont très proches, indiquant que les significations correspondantes sont voisines [Bianco05].

Lors l'utilisation de la LSA dans le cadre d'indexation elle devient la LSI [Deerwester90], cette technique considère que deux mots apparaissant fréquemment dans le même contexte, même s'ils n'apparaissent jamais dans les mêmes documents, auront le même sens et seront donc considérés comme des synonymes. Réciproquement, un mot apparaissant fréquemment dans deux contextes différents aura deux sens différents et sera donc considéré comme polysémique.

Cette technique non-supervisée est statistique, elle est basée sur une approximation de la matrice mot clé-document par une matrice de rang inférieur.

5.5.1. Décomposition en valeurs singulières

La théorie sur laquelle s'appuie LSI est la décomposition en valeurs singulières (SVD). Une matrice $A = [a_{ij}]$ où a_{ij} est la fréquence d'apparition du mot i dans le contexte j, se décompose en un produit de trois matrices USV^T. U et V sont des matrices orthogonales et S une matrice diagonale. La **Figure 6** représente le schéma bien connu d'une telle décomposition où r représente le rang de la matrice A (voir Annexe).

Soit S_k où $k < r$ la matrice produite en enlevant de S les $r - k$ colonnes qui ont les plus petites valeurs singulières. Soit U_k et V_k les matrices obtenues en enlevant les colonnes correspondantes des matrices U et V . La matrice $U_k S_k V_k^T$ peut alors être considérée comme une version compressée de la matrice originale A. Le but de la réduction de la dimension est de réduire le « bruit » dans l'espace latent, résultant en une structure de relation de mot plus riche qui révèle la sémantique latente présents dans la collection.

Avant d'effectuer la décomposition en valeurs singulières, il faut normaliser la matrice d'origine A. Cette normalisation consiste à appliquer un logarithme sur la matrice A. Ainsi, plutôt que de se fonder directement sur le nombre d'occurrences de chacun des mots, une telle transformation permet de s'appuyer sur une estimation de l'importance de chacun des mots dans leur contexte. De manière similaire aux travaux de Turney, cette étape de normalisation peut également s'appuyer sur la méthode du Tfidf [Roche06].

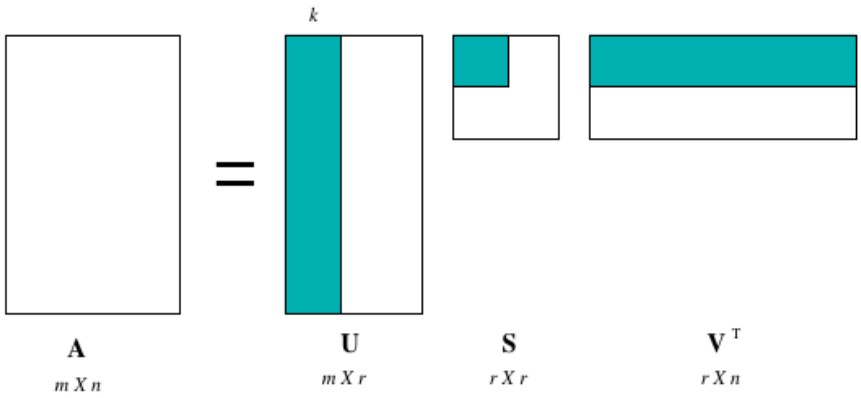

Figure 6: Décomposition en valeurs singulières. La matrice A représente le corpus d'origine de m lignes (mots du corpus) et n colonnes (contextes).

LSI s'appuie sur le paramètre *k*, pour la réduction de la dimension. Le nombre adéquat de dimensions ne peut pas être actuellement déterminé théoriquement ; seuls des tests empiriques ont permis de situer cette valeur entre 100 et 300 dans le cas de l'anglais [Deerwester90]. En général, les valeurs de *k* les plus petites sont préférés lors de l'utilisation de la LSI, en raison du coût de calcul associé à l'algorithme SVD, ainsi que le coût du stockage et la comparaison des vecteurs de grandes dimensions [Kontostathis06].

6. Structure des fichiers index

La structure d'index est la manière dont l'index doit être organisé (stocké) pour répondre efficacement et rapidement aux besoins des utilisateurs. Les structures d'index, à elles seules, déterminent la méthode d'indexation et agissent directement sur les performances du système de recherche d'information. Plusieurs études ont été menées sur les structures de fichier utilisées dans la recherche d'information [Rijsbergen80]. Une structure de fichiers idéale doit être [Dahak06]:

- rapide pour la recherche d'un élément ;
- rapide pour la mise à jour de l'indexation ;
- permettre la manipulation de gros fichiers de données ;
- facile à mettre en œuvre ;
- occuper le moins de place par rapport à celle occupée par les fichiers de données.

Malheureusement aucune structure de fichier ne respecte tous ces critères en même temps.

Nous présentons dans ce qui suit quelques structures utilisées dans le domaine de la RI.

6.1. Les fichiers Séquentiels (Sequential files)

Un fichier séquentiel est le moyen le plus simple de stocker un fichier de données puisque l'on stocke les enregistrements les uns à la suite des autres dans leur ordre d'insertion. Jusqu'au milieu des années 70, tous les systèmes textuels utilisaient les fichiers séquentiels du fait de l'utilisation de bandes magnétiques. Une requête sur un document consistait alors à parcourir toute la bande jusqu'à ce que l'on trouve le bon document, d'où une lenteur certaine du système malgré l'optimisation des algorithmes de recherche.

6.2. Les fichiers inversés (Inverted files)

La structure de fichier inversé est à la base de tous les systèmes de recherche d'information. Un système à base de fichiers inversés contient trois composants principaux :

- **Un dictionnaire** : Le fichier dictionnaire contient tous les mots ou groupes nominaux spécifiques pouvant servir de mots-clés pour l'indexation et la recherche dans l'ensemble des fichiers à traiter. A chaque entrée du dictionnaire est associé le nombre de fois où l'entrée apparaît dans l'ensemble documentaire.
- **Un fichier de hachage** : Ce fichier contient pour chaque entrée du dictionnaire une liste décrivant dans quel fichier apparaît cette entrée. Cette méthode permet de restreindre l'étude sur les fichiers qui nous intéressent et pas les autres.
- **Les fichiers de données :** Représentent les documents du corpus à indexer.

La **Figure 7** représente un système à base de fichiers inversés.

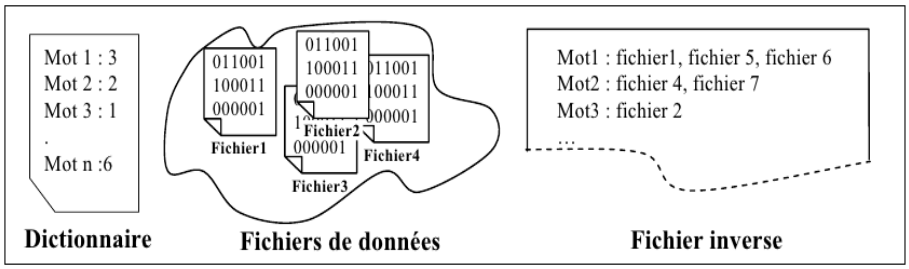

Figure 7: Système à base de fichiers inversés.

6.3. Les fichiers de signature (Signature files)

Les limitations des fichiers inversés ont amené à revoir les fichiers séquentiels pour les adapter

à une représentation efficace dans tous les domaines. Une signature de texte est une chaîne de bits (chaîne binaire) dans laquelle les bits sont positionnés pour décrire le document. La chaîne de bit est créée en appliquant une opération de hachage sur tous les mots clés décrivant un document. Une fois qu'un fichier signature est créé, la réponse à une requête consiste à calculer une chaîne pour la requête et à la comparer à celle calculée pour chaque document. Les manipulations de bits étant codées dans le processeur, cette technique est beaucoup plus rapide que la manipulation de chaînes de caractères. Cette méthode est donc très rapide, bien que le fichier signature ait besoin d'être parcouru entièrement à chaque requête. Le principal désavantage de cette méthode est que le fait qu'un bit signale la présence d'un mot dans un fichier ne donne ni son emplacement, ni le nombre de fois qu'il apparaît dans ce fichier.

6.4. Les fichiers multilistes (Multi-lists files)

La structure de fichiers multi-liste est à mi-chemin entre les fichiers séquentiels et les fichiers inversés. Elle est seulement un fichier inversé légèrement modifié. Il y a une liste par mot-clé. Les enregistrements contenant un mot-clé particulier K_i sont enchaînés ensemble pour former la K_i-liste et on retourne le début de la K_i-liste dans la liste d'adresses, comme illustré dans la **Figure 8**. Un fichier multi-liste contient :

- Un fichier dictionnaire : Analogue au fichier de hachage décrit pour le système de fichiers inversés. Chaque entrée dans le fichier pointe vers l'emplacement du mot dans le fichier. Cependant, seul le premier fichier contenant le mot est stocké.
- Les fichiers de données : Chaque mot dans un document possède un pointeur vers le prochain document contenant ce mot. Une recherche de tous les documents contenant le mot-clé ***M***, revient à regarder dans le fichier dictionnaire où se trouve le premier fichier contenant ce mot, puis à parcourir les documents tant qu'il y a pour ce mot-clé ***M*** un pointeur vers un autre document. Cette méthode est rapide pour la mise à jour des fichiers, mais elle est très gourmande en accès disque car on lit des fichiers multiples à des endroits différents du disque.

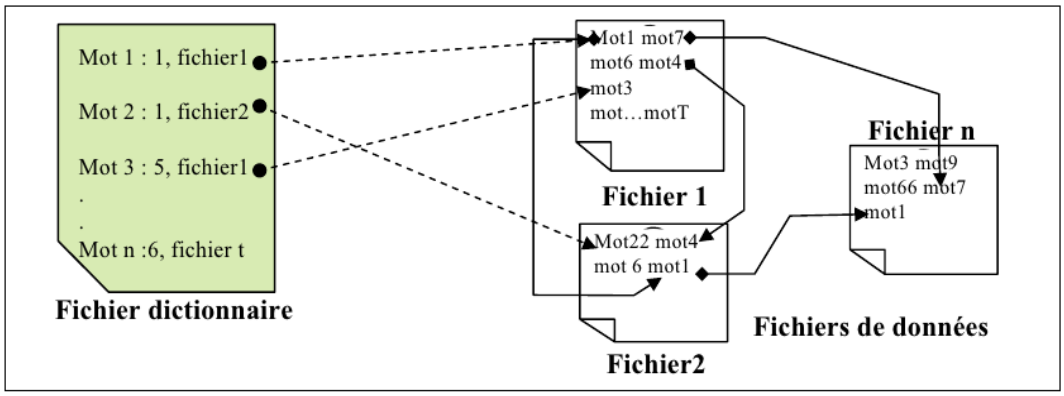

Figure 8: Structure de fichier multi-liste.

7. Similarité

En recherche d'information, le problème principal est d'évaluer les similarités entre les éléments stockés dans une base documentaire et des requêtes représentant les besoins d'information exprimés par les utilisateurs. Calculer la similarité entre deux documents, ou bien entre un document et une requête, est une manière de voir combien proches les documents (respectivement le document et la requête) le sont. Autrement dit, plus la distance entre les deux est petite, plus la similarité est grande; donc plus ils sont similaires.

L'étude de la similarité entre les mots a fait l'objet de nombreuses recherches [Bannour11] Lee analyse certaines mesures de similarité en prenant en considération leurs propriétés formelles. Lin décrit et compare un ensemble de mesures de similarité. Weeds et Weir évaluent les mesures de similarité proposées par Lee et Lin en termes de rappel et de précision. Strehl effectue une comparaison détaillée de mesures et étudie leur influence sur la classification ultérieure des mots (clustering). Parmi les algorithmes de similarités les plus utilisées :

7.1. Cosinus

La similarité cosinus permet de calculer la similarité entre deux vecteurs de dimensions *n* en déterminant l'angle entre eux. Elle est issue de l'algèbre linéaire et elle est devenue plutard la mesure standard des vecteurs pondérés dans le domaine de la recherche d'information. Witten [Bannour11] préconise l'utilisation de cette mesure dans le domaine de la recherche d'information plutôt que le produit scalaire.

Soit deux vecteurs *A* et *B*, la similarité est :

$$similarité(A,B)=\cos(\theta)=\frac{A\cdot B}{\|A\|\|B\|}=\frac{\sum_{i=1}^{n}(A_i\times B_i)}{\sqrt{\sum_{i=1}^{n}(A_i)^2}\times\sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

Formule 9: Cosinus similarité entre le vecteur A et B

En règle générale, pour mesurer finement la similarité entre des séquences de texte, les vecteurs sont construits d'après un calcul de type Tfidf.

7.2. Coefficient de Jaccard

L'indice de Jaccard appliqué à un texte correspond aux rapport entre le nombre de mots présent dans les deux textes et le nombre de mots des deux textes. L'indice de Jaccard est normalisé (entre 0 et 1), plus il est proche de 1 (ou 100%) plus les deux textes sont similaires [Jaccard1901]. Soit deux ensembles A et B, l'indice est :

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

Formule 10:
Jaccard

7.3. Indice de Sørensen

L'indice Sørensen est utilisée pour comparer la similitude de deux échantillons. Il a été développé par le botaniste Sørensen Thorvald [Sørensen48].

La formule originale de Sørensen est destiné à être appliqué à la présence/absence de données :

$$QS=\frac{2C}{A+B}=\frac{2n(A\cap B)}{n(A)+n(B)}$$

Formule 11: Sørensen

Où *n(A)* et *n(B)* sont le nombre d'espèces dans les échantillons *A* et *B*, respectivement, et *C* est le nombre d'espèces partagées par les deux échantillons; *QS* est le quotient de similarité et varie entre 0 et 1.

7.4. Distance de Jaro-Winkler

La distance de Jaro-Winkler mesure la similarité entre deux chaînes de caractères. Il s'agit d'une variante proposée en 1999 par William E. Winkler, découlant de la distance de Jaro (1989, Matthew A. Jaro) qui est principalement utilisée dans la détection de doublons [Winkler99].

Plus la distance de Jaro-Winkler entre deux chaînes est élevée, plus elles sont similaires. Cette mesure est particulièrement adaptée au traitement de chaînes courtes comme des noms ou des mots de passe. Le résultat est normalisé de façon à avoir une mesure entre 0 et 1, le zéro représentant l'absence de similarité.

7.4.1. Distance de Jaro

La distance de Jaro entre les chaînes A et B est définie par :

$$dj = \frac{1}{3}\left(\frac{m}{|A|} + \frac{m}{|B|} + \frac{m-t}{m}\right)$$

Formule 12: Distance de Jaro

où:

- $|A|$ et $|B|$ est la longueur de la chaîne de caractères A et B;
- m est le nombre de *caractères correspondants*;
- t est le nombre de *transpositions*.

Deux caractères identiques de A et de B sont considérés comme *correspondants* si leur éloignement (la différence entre leurs positions dans leurs chaînes respectives) ne dépasse pas :

$$\left\lfloor \frac{max(|A|,|B|)}{2} \right\rfloor - 1$$

Le nombre de transpositions est obtenu en comparant le i-ème caractère *correspondant* de A avec le i-ème caractère *correspondant* de B. Le nombre de fois où ces caractères sont différents, divisé par deux, donne le nombre de *transpositions*.

7.4.2. Distance de Jaro-Winkler

La méthode introduite par Winkler utilise un *coefficient de préfixe P* qui favorise les chaînes commençant par un préfixe de longueur l (avec $l \leq 4$). En considérant deux chaînes A et B, leur distance de Jaro-Winkler d_w est :

$$d_w = d_j + (lp(1 - d_j))$$

Formule 13: Jaro-Winkler

où :

- d_j est la distance de Jaro entre *A* et *B ;*
- *l* est la longueur du préfixe commun (maximum 4 caractères) ;
- *p* est un coefficient qui permet de favoriser les chaînes avec un préfixe commun. Winkler propose pour valeur p = 0,1.

7.5. Distance de Levenshtein

La distance de Levenshtein mesure la similarité entre deux chaînes de caractères. Elle est égale au nombre minimal de caractères qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre. Son nom provient de Vladimir Levenshtein qui l'a définie en 1965 [Levenshtein66].

On appelle distance de Levenshtein entre deux mots *M* et P le coût minimal pour aller de *M* à *P* en effectuant les opérations élémentaires suivantes:

- substitution d'un caractère de *M* en un caractère de *P*.
- ajout dans *M* d'un caractère de *P*.
- suppression d'un caractère de *M*.

On associe ainsi à chacune de ces opérations un coût. Le coût est toujours égal à 1, sauf dans le cas d'une substitution de caractères identiques. Exemples :

- Si *M* = « examen » et *P* = « examen », alors LD(M,P) = 0, parce qu'aucune opération n'a été réalisée.
- Si *M* = « exam**e**n » et *P* = « exam**a**n », alors LD(M,P) = 1, parce qu'il y a eu un remplacement (changement du *e* en *a*).
- Si *M* = « exame » et *P* = « exame**n** », alors LD(M,P) = 1, parce qu'il y a eu un remplacement (ajout du *n* a la fin).

8. La classification

La classification de textes a pour objectif de regrouper les textes similaires, c'est à dire thématiquement proches, au sein d'un même ensemble. L'intérêt d'une telle démarche est d'organiser les connaissances de façon à pouvoir effectuer, par la suite, une recherche ou une extraction d'information efficace [Jaillet03].

On distingue dans le domaine de la classification automatique deux types d'approches : la classification supervisée et la classification non supervisée. Ces deux méthodes diffèrent sur la façon dont les classes sont générées. En effet dans le cas de la classification non supervisée, les groupes de documents (classes) sont calculés automatiquement par la machine, tandis qu'ils sont, dans l'approche supervisée, définis par un expert. Dans ce dernier cas, il est intéressant de représenter les documents et les classes à l'aide d'un même formalisme et celui généralement utilisé est un espace vectoriel [Jaillet03].

Dans notre cas, nous nous intéresserons à la catégorisation, c'est à dire aux algorithmes d'apprentissage supervisés et plus particulièrement aux méthodes basées sur une représentation vectorielle de documents. Le modèle vectoriel standard est issu de Salton en 1971 dont l'implémentation la plus connue est SMART [Jaillet03]. Dans ce formalisme, chaque dimension de l'espace vectoriel correspond à un élément textuel, nommé terme d'indexation, préalablement extrait du jeu d'apprentissage. Parmi les méthodes reconnues comme les plus performantes : k plus proches voisins, machine à vecteur de support, naïve bayes, arbres de décision, réseau de neurones (voir Annexe).

8.1. Les k plus proches voisins (k-NN)

k-NN (k-Nearest Neighbour) est une méthode très connue dans le domaine de la catégorisation automatique. Ses performances la situe parmi les meilleures méthodes de catégorisation dans [Jaillet03]. L'idée de k-NN est de représenter chaque texte dans un espace vectoriel, dont chacun des axes représente un élément textuel[3].

L'algorithme KNN figure parmi les plus simples algorithmes d'apprentissage artificiel. Dans un contexte de classification d'une nouvelle observation x, l'idée fondatrice simple est de faire voter les plus proches voisins de cette observation. La classe de x est déterminée en fonction de la classe majoritaire parmi les k plus proches voisins de l'observation x.

3 Un élément textuel peut être un mot sous sa forme "brute" ou sous une forme lemmatisée etc..

8.1.1. Algorithme KNN

La méthode du plus proche voisin est une méthode non paramétrique où une nouvelle observation est classée dans la classe d'appartenance de l'observation de l'échantillon d'apprentissage qui lui est la plus proche, au regard des covariables utilisées. La détermination de leur similarité est basée sur des mesures de distance. Formellement, soit *L* l'ensemble de données à disposition ou échantillon d'apprentissage :

$L = \{(y_i, x_i),\ i=1,\ldots,\ n_L\}$

où $y_i \in \{1,\ldots,c\}$ dénote la classe de l'individu i et le vecteur $x_i=(x_{i1},\ldots,\ x_{ip})$ représente les variables prédictrices de l'individu *i*. La détermination du plus proche voisin est basée sur un fonction distance arbitraire *d*(.,.). La distance euclidienne ou dissimilarité entre deux individus caractérisés par *p* covariables est définie par:

$$d((x_1, x_2, \ldots, x_p), (u_1, u_2, \ldots, u_p)) = \sqrt{(x_1-u_1)^2+(x_2-u_2)^2+\ldots+(x_p-u_p)^2}$$

Ainsi, pour une nouvelle observation (y,x) le plus proche voisin $(y_{(1)},x_{(1)})$ dans l'échantillon d'apprentissage est déterminé par :

$$d(x, x_{(1)}) = \min_i(d(x, x_i))$$

et $\hat{y} = y_{(1)}$, la classe du plus proche voisin, est sélectionnée pour la prédiction de y. Les notations $x^{(j)}$ et y(j) représentent respectivement le $j^{ème}$ plus proche voisin de x et sa classe d'appartenance.

Parmi les fonctions distance types, la distance euclidienne est définie comme suit :

$$d(x_i, x_j) = \sum_{s=1}^{p} ((x_{is}-x_{js})^2)^{\frac{1}{2}}$$

Formule 14: Distance euclidienne

et plus généralement la distance de Minkowski :

$$d(x_i, x_j) = \sum_{s=1}^{p} (|x_{is}-x_{js}|^q)^{\frac{1}{q}}$$

Formule 15: Distance de Minkowski

9. Conclusion

Nous avons présenté dans ce chapitre les principales notions et concepts de la recherche d'information. Nous avons développé les principales étapes du processus de recherche d'information qui sont la représentation, l'indexation de l'information et la comparaison de l'information et du besoin en information.

Dans les systèmes classiques de recherche d'informations, l'indexation est organisée en trois étapes : l'extraction, la sélection et la pondération des termes. L'objectif final est de définir pour chaque document ses termes d'indexation.

Nous avons introduit la notion de similarité, qui a pour but de trouver dans un ensemble de document le sous groupe de ceux qui sont similaires à la requête. Les modèles de RI permettent la représentation et la comparaison des représentations des documents et des requêtes. Ils utilisent généralement des mesures de distance basées sur les attributs des objets comparés. Les méthodes de classifications les plus utilisés dans le domaine de la RI sont présentées.

Chapitre 3: Conception et Réalisation

1. Introduction

Plusieurs travaux de recherches ont été menés dans plusieurs domaines liés à la technologie de l'indexation et études linguistiques, notamment les méthodes de pondérations, le traitement des requêtes, les techniques de compressions et de stockage de la base de données et la classification de nouveau documents. Pour tester ces méthodes, nous avons été amené à étudier et à implémenter un indexeur réel.

2. Corpus utilisé

Nous avons utilisé le corpus que nous avons établie dans le cadre de notre projet de fin d'étude [Roukh10] pour nos tests.

Ce corpus comporte 438.241 documents et plusieurs catégories : politique, actualité du monde, économie, sport, art et culture, science, etc.

Nous avons utilisé ce corpus pour expérimenter ces méthodes d'indexation.

3. Modélisation UML

UML (Unified Modeling Language, que l'on peut traduire par "langage de modélisation unifié) est une notation permettant de modéliser un problème de façon standard. Ce langage est né de la fusion de plusieurs méthodes existant auparavant, il est devenu désormais la référence en terme de modélisation objet.

3.1. Diagramme de cas d'utilisations

Le diagramme de cas d'utilisation est utilisé pour donner une vision globale du comportement fonctionnel d'un système logiciel. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Il est une unité significative de travail. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs (actors), ils interagissent avec les cas d'utilisation (use cases).

UML définit une notation graphique pour représenter les cas d'utilisation, cette notation est appelée diagramme de cas d'utilisation. UML ne définit pas le standard pour la forme écrite de ces cas d'utilisation, et en conséquence il est aisé de croire que cette notation graphique suffit à elle seule pour décrire la nature d'un cas d'utilisation. Dans les faits, une notation graphique peut

seulement donner une vue générale simplifiée d'un cas ou d'un ensemble de cas d'utilisation. Les diagrammes de cas d'utilisation sont souvent confondus avec les cas d'utilisation. Bien que ces deux concepts soient reliés, les cas d'utilisation sont bien plus détaillés que les diagrammes de cas d'utilisation.

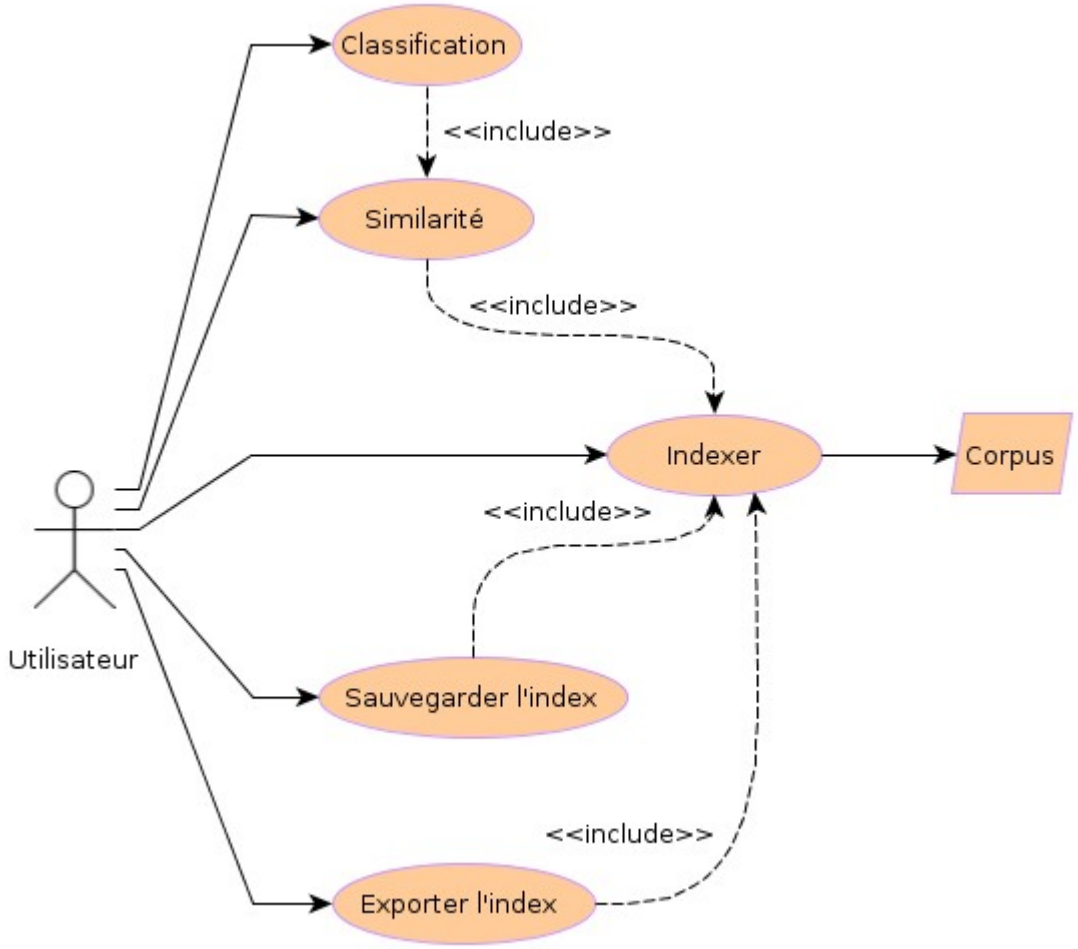

Figure 9: Diagramme UML cas d'utilisations.

L'acteur principal est l'utilisateur, il peut indexer, classifier ou rechercher une requêtes depuis le corpus.

3.2. Diagramme d'activité

Un diagramme d'activité permet de modéliser un processus interactif, global ou partiel pour un système donné (logiciel, système d'information). C'est une représentation proche de l'organigramme; la description d'un cas d'utilisation par un diagramme d'activités correspond à sa

traduction algorithmique. Une activité est l'exécution d'une partie du cas d'utilisation, elle est représentée par un rectangle aux bords arrondis.

Nous allons examiner ci-après à l'aide du diagramme d'activité comment nous avons structuré notre moteur d'indexation. Cette phase schématise toutes les analyses faites dans le chapitre 2.

3.2.1. Extraction des termes (tokenisation)

La première étape du processus d'indexation consiste à segmenter le texte en un ensemble d'unités lexicales élémentaires, prenant généralement la forme de mots. Le traitement associé est la tokenisation. Malgré sa simplicité, tout le processus d'indexation syntaxique repose sur l'extraction des termes . Elle est directement influencée par le langage dans lequel le document est écrit.

3.2.2. Suppression des mots vides

Ce traitement consiste à éliminer tous les mots peu informatifs tels que les prépositions, les articles, les pronoms ainsi que les auxiliaires. Les termes issus d'une liste de mots vides sont supprimés. Nous avons utilisé une liste comportant 637 termes pour le français que nous avons déjà considérée dans le cadre de TREC [Lien01] et une liste de 571 termes pour l'anglais issue du projet SMART [Lien02].

3.2.3. Lemmatisation

Nous avons procédé à la phase classique de lemmatisation des termes et nous avons utilisé TreeTagger [Schmid94]. C'est un outil qui permet l'étiquetage morpho-syntaxique et la lemmatisation. Il a été développé par Helmut Schmid dans le cadre de projet TC dans l'ICLUS (Institut for Computational Linguistics of the University of Stuttgart) [Schmid94]. Cet outil est gratuit, disponible en ligne et facile à installer sur les systèmes d'exploitations Linux et Windows. Il a été utilisé avec succès pour étiqueter de nombreuses langues (anglais, français, allemand, italien, néerlandais, espagnol, bulgare, russe, grec, portugais, chinois, etc.).

3.2.4. Stemmatisation

Les termes restants sont radicalisés afin de limiter les variations syntaxiques. Nous avons utilisé pour cette étape l'algorithme de Porter [Porter80], cet algorithme permet de supprimer les affixes des mots pour obtenir une forme canonique. Il est composé d'une cinquantaine de règles de racinisation/désuffixation classées en sept phases successives (traitement des pluriels et verbes à la

troisième personne du singulier, traitement du passé et du progressif, etc.) [Porter80].

3.2.5. Pondération

Les termes sont pondérés par leur importance dans des relations avec les termes dans le même document et les autres documents. Le poids d'un terme est une combinaison entre les pondérations locale, globale et la normalisation fournis par l'algorithme TFIDF [Salton88]. L'algorithme LSI permet de résoudre les problèmes de synonymie et polysémie présent dans le modèle vectorielle [Deerwester90].

3.2.6. Sauvegarde de l'index

Les termes pondérés (descripteur ou index) associés à chaque document doivent être représentés dans des structures de stockage destinées à faciliter leur accès. Ces structures vont être utilisées par le SRI lors de la phase de recherche. Nous avons utilisé la technique des fichiers inverses sous format XML pour l'organisation des données, ces derniers contiennent les différentes informations concernant tous termes du corpus. Les métas données présentes pour chaque document sont :

1. **Type d'index:** TFIDF ou LSI.
2. **Path:** le chemin de corpus dans l'ordinateur.
3. **Le terme:** Le terme indexé.
4. **La fréquence du terme:** dans les documents du corpus.
5. **L'identificateur de document:** l'id du document ou il appartient le terme.
6. **La valeur de pondération:** l'importance du terme dans le corpus après les phases d'indexation (TFIDF ou LSI).

```
<?xml version="1.0" encoding="UTF-8"?>
<tfidf numDocs="78" path="/home/amine27/Developpement/C++/storiesindexer/corpus/afp-en/business">
    <item term="watch" docFreq="4">
        <doc id="0">0.594083</doc>
        <doc id="10">0</doc>
        <doc id="22">0</doc>
        <doc id="42">0.247535</doc>
        <doc id="45">0.34274</doc>
        <doc id="50">0.742604</doc>
        <doc id="55">0</doc>
    </item>
    <item term="index" docFreq="5">
        <doc id="0">1.09891</doc>
        <doc id="18">0.457878</doc>
        <doc id="22">0.549454</doc>
        <doc id="50">0.686818</doc>
        <doc id="66">0.249752</doc>
    </item>
    <item term="occupi" docFreq="1">
        <doc id="77">0.544589</doc>
    </item>
    <item term="manhattan" docFreq="1">
        <doc id="77">0.544589</doc>
    </item>
    <item term="instinet" docFreq="1">
        <doc id="77">0.544589</doc>
    </item>
</tfidf>
```

Figure 10: Extrait de l'index final (format xml).

Nous avons ajouté la possibilité de sauvegarder l'index réduit des termes les plus fréquents dans le corpus (10 termes depuis chaque document).

3.2.7. Traitement des requêtes

L'interrogation ou le traitement des requêtes permet de comparer une question donnée à tous les documents de la base. La pondération du résultat de cette comparaison permet de ranger les réponses dans un ordre décroissant de pertinence. Plusieurs algorithmes de similarité on été implémenté dans ce sens : cosinus, jaccard, sorensen, jaro winkler, levenshtein et les n-grams.

3.2.8. Classification de nouveaux documents

La classification automatique de documents consiste à associer une catégorie à un document pouvant être une phrase, un paragraphe, un texte, etc. Pour cette tâche, nous avons choisi d'implémenter un algorithme classique de classification de données textuelles, les k plus proches voisins (k-ppv). Le principe de cet algorithme est de mesurer la similarité entre un nouveau document et l'ensemble des documents déjà classés. Les méthodes de similarité que nous utilisé sont : jaccard, sorensen, jaro winkler, levenshtein et les n-grams.

Nous avons utilisé pour l'apprentissage 100 documents dans chaque catégorie extrait depuis le corpus principal. Pour le teste 7 documents dans chaque catégorie.

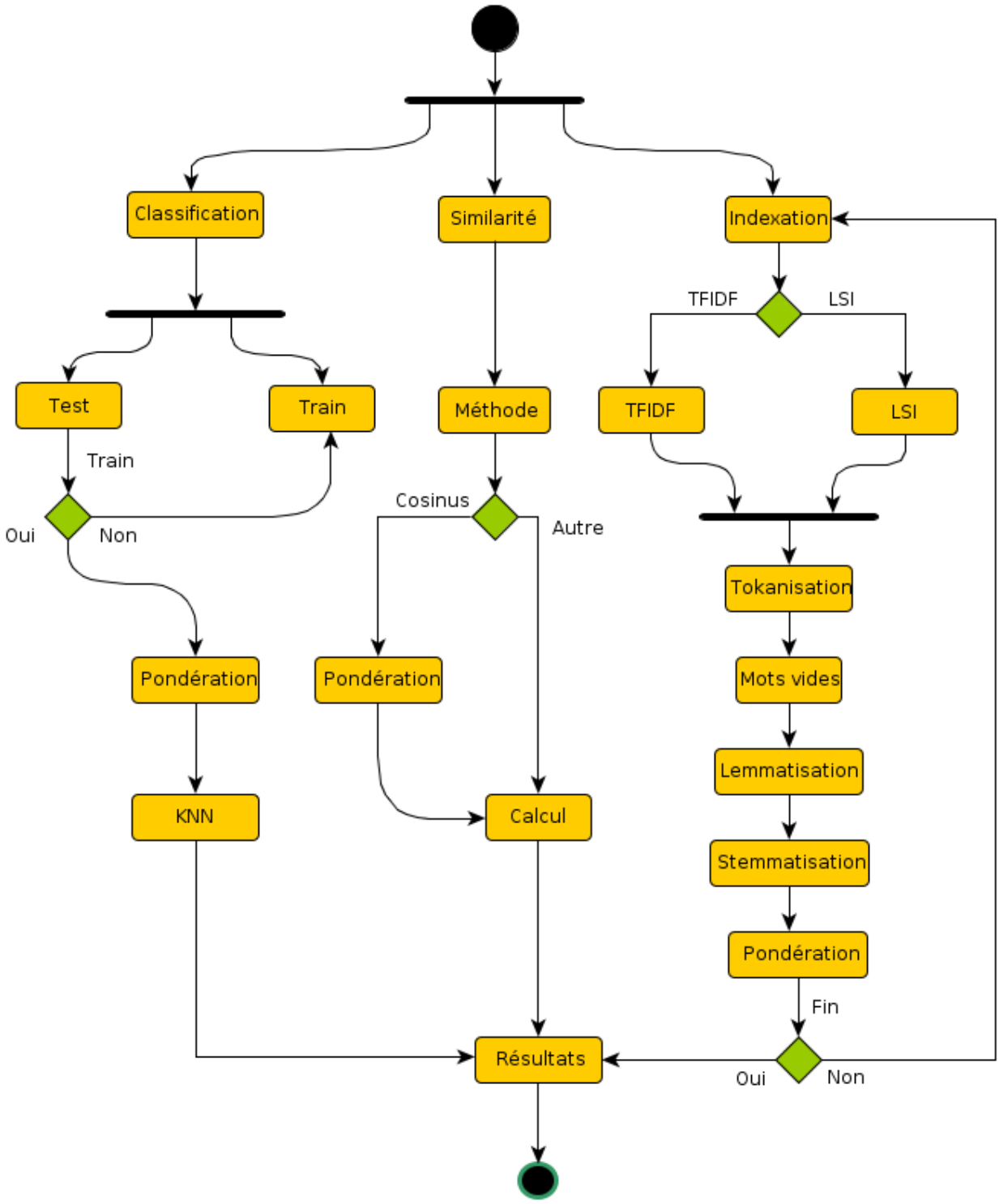

Figure 11: Diagramme UML d'activité.

3.3. Diagramme de classes

Le diagramme de classes est un schéma utilisé en génie logiciel pour présenter les classes et les

interfaces des systèmes ainsi que les différentes relations entre celles-ci. Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe. Une classe est un ensemble de fonctions et de données (attributs) qui sont liées ensembles par un champ sémantique. Les classes sont utilisées dans la programmation orientée objet. Elles permettent de modéliser un programme et ainsi de découper une tâche complexe en plusieurs petites taches simples. Notre diagramme de classes :

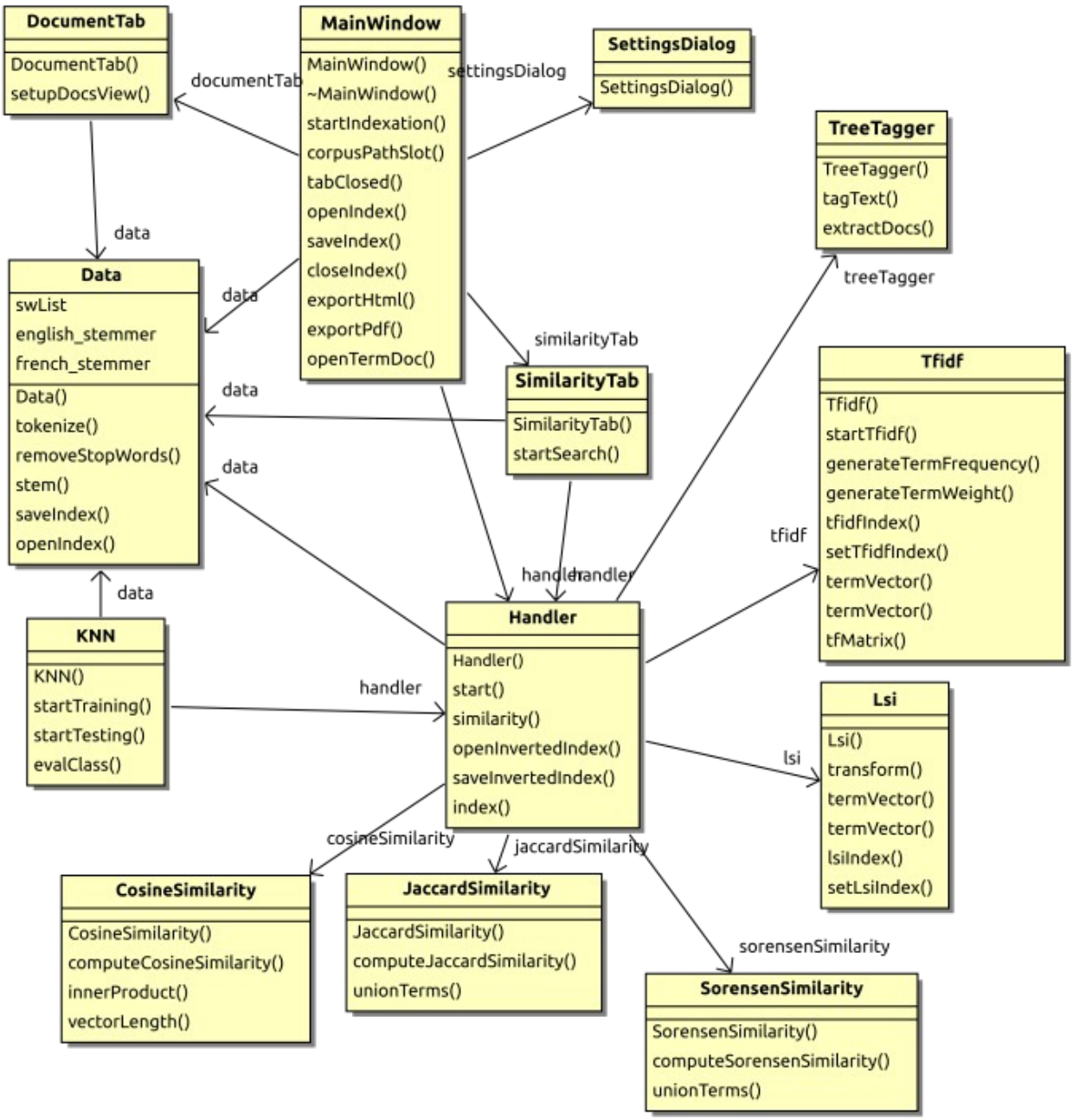

Figure 12: Diagramme UML de classe.

Dans notre projet, nous avons implémenté les classes suivantes:

MainWindow: la classe principale de l'interface utilisateur permet d'afficher et de faire appel aux classes : **DocumentTab** et **SimilarityTab** et **SettingsDialog**.

Handler: la classe qui gère les différentes taches de l'application, tel que l'indexation la recherche d'une requête et la classification.

Tfidf, LSI : implémente les algorithmes d'indexation Tfidf et LSI.

CosineSimilarity, **JaccardSimilarity**, **SorensenSimilarity**: implémente les algorithmes de similarité.

KNN: implémente l'algorithme de classification k-plus proche voisins.

TreeTagger: grère l'interface avec le processus tree-tagger.

Data: permet d'écrire ou lire les document du corpus ou le fichier inverse, ainsi que la tokenisation et la stemmatisation.

Les relations sont de type agrégation (utilisation), chaque classe concernée utilise un objet de l'autre classe.

4. Environnement de programmation

Le système d'exploitation utilisé dans notre projet est GNU/Linux sous la distribution Kubuntu 11.10 et le gestionnaire de bureau KDE 4. Le langage de programmation et les technologies utilisés sont:

4.1. C++

Le C++ est un langage de programmation permet la programmation sous de multiples paradigmes comme la programmation procédurale, la programmation orientée objet et la programmation générique. C++ est actuellement le 2e langage le plus utilisé dans le monde. Le langage C++ n'appartient à personne et par conséquent n'importe qui peut l'utiliser sans avoir besoin d'une autorisation ou obligation de payer pour avoir le droit d'utilisation. Nous avons choisi ce langage parce qu'il est très rapide en exécution (assez bas niveau) et portable.

4.2. Qt

La bibliothèque Qt est un framework orienté objet développé en C++ par Qt Development

Frameworks, filiale de Nokia. Elle offre des composants d'interface graphique (widgets), d'accès aux bases de données, de connexions réseaux, de gestion des fils d'exécution, d'analyse XML, la 3D par OpenGL, le Javascript (ECMAScript), des outils d'internationalisation, etc. Qt est par certains aspects un framework lorsqu'on l'utilise pour concevoir des interfaces graphiques dont leur architecture utilise les mécanismes des signaux et slots par exemple.

Figure 13: Qt Logo.

Qt permet la portabilité des applications qui n'utilisent que ses composants par simple recompilation du code source. Les environnements supportés sont les Unix (dont Linux) qui utilisent le système graphique X Window System, Windows et Mac OS X. Le fait d'être une bibliothèque logicielle multiplate-forme attire un grand nombre de personnes qui ont donc l'occasion de diffuser leurs programmes sur les principaux OS existants. Qt est libre et open-source sous la licence GNU/GPL et GNU/LGPL ainsi qu'une version commerciale.

Qt est notamment connu pour être la bibliothèque sur laquelle repose l'environnement graphique KDE, l'un des environnements de bureau les plus utilisés dans le monde Linux, elle est aussi utilisée pour la création de nombreuses applications comme : Google Earth, Opera, OPIE, VoxOx, Skype, VLC media player, Autodesk Maya et VirtualBox...etc.

4.3. Qt Creator

Qt Creator est un environnement de développement intégré multiplate-forme faisant partie du framework Qt. Il intègre directement dans l'interface un débogueur, un outil de création d'interfaces graphiques ainsi que la documentation Qt. L'éditeur de texte intégré permet l'auto-complétion ainsi que la coloration syntaxique, un éditeur de code C++ avancé, des outils de gestion de version, de projet, de code et des outils de navigation. Qt Creator utilise sous Linux le compilateur gcc et MinGW par défaut sous Windows.

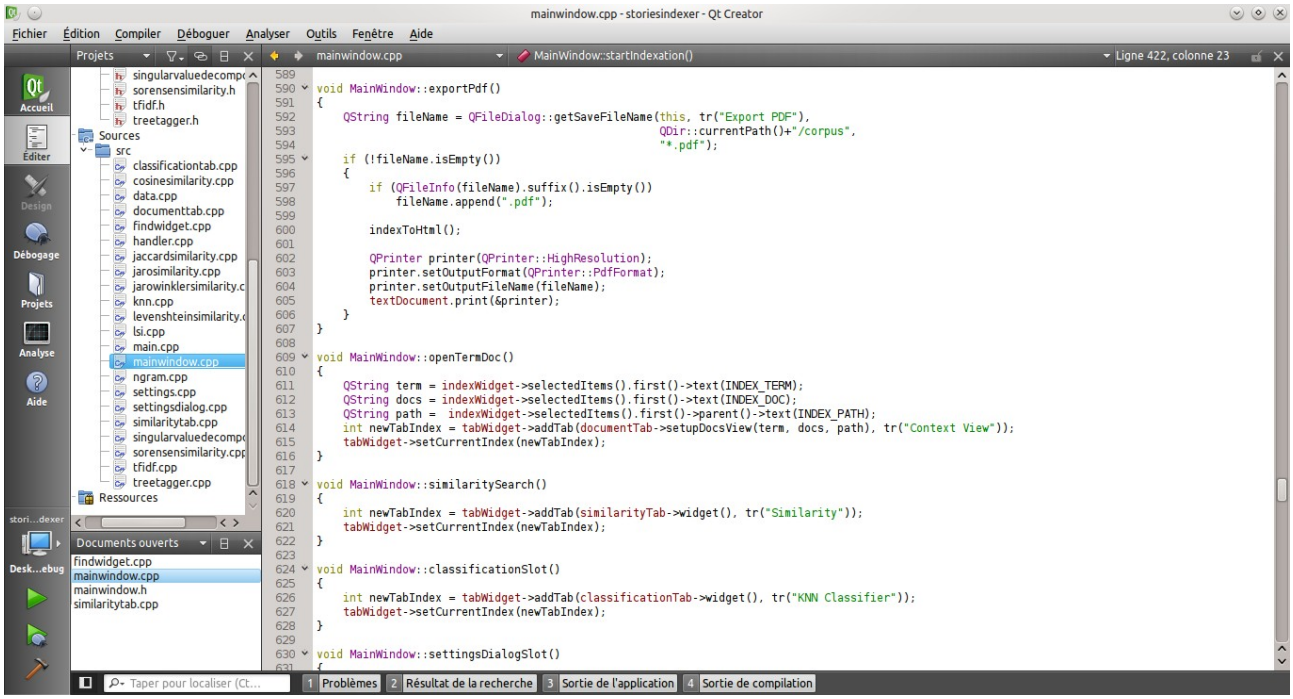

Figure 14: Qt Creator.

5. Interface utilisateur

5.1. Première fenêtre

La première fenêtre de l'application contient quatre onglets principaux, un onglet qui permet de lancer et d'afficher l'indexation du corpus, le deuxième onglet permet d'afficher le document qui contient le terme sélectionné, le troisième onglet permet de calculer la similarité entre le corpus indexé et une requête donnée, le dernier onglet permet de classifier un nouveau document à l'aide de l'algorithme *kPPV*.

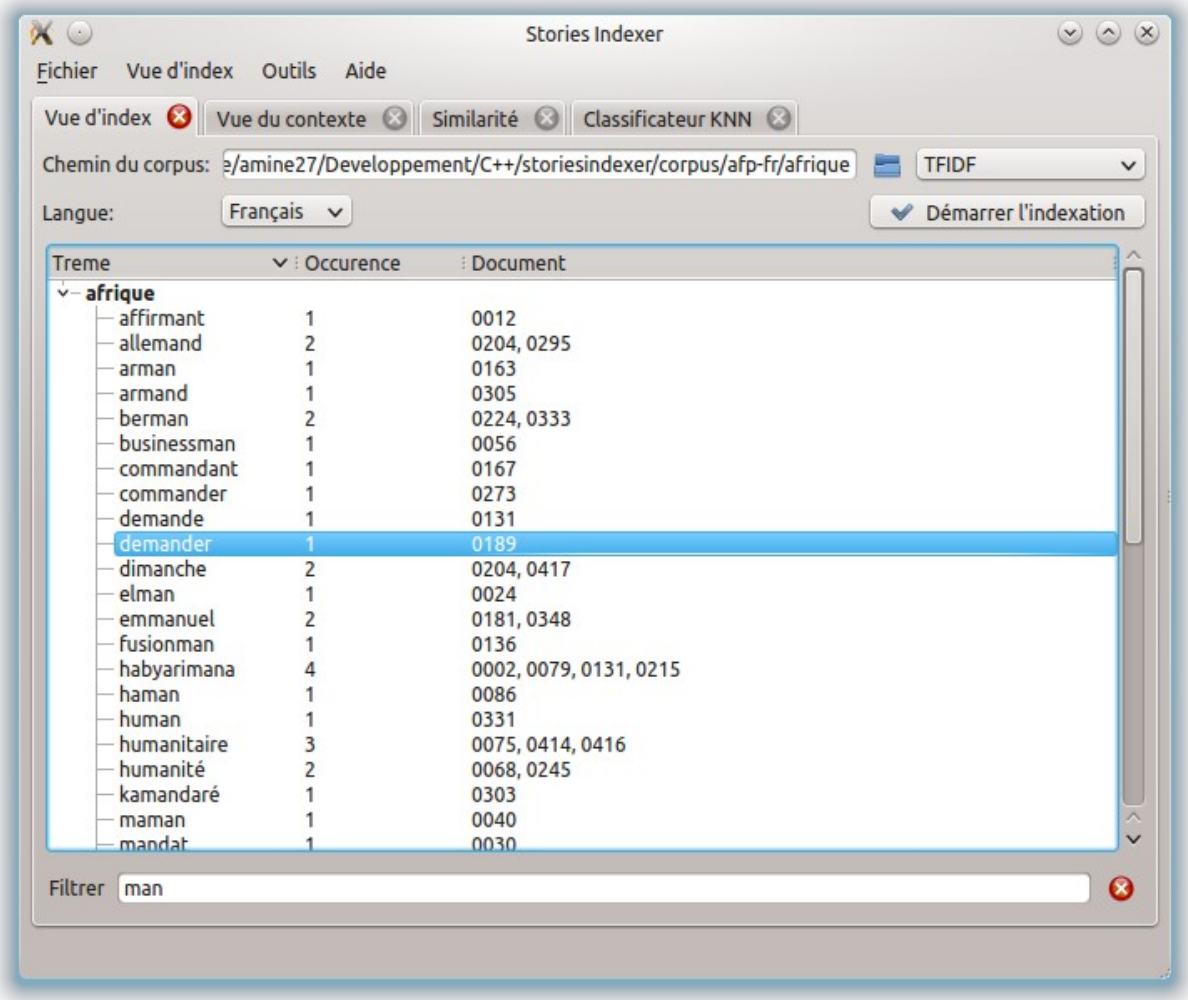

Figure 15: Fenêtre principale.

5.2. Vue d'index

Elle permet de faire l'indexation, on choisit le chemin du corpus, la méthode et le langage d'indexation, puis on clique sur Démarrer l'indexation. On peut filtrer des mots dans l'index grâce a la barre de filtre.

5.3. Vue du contexte

Elle permet d'afficher les documents qui contiennent le terme sélectionné dans la vue d'index avec la possibilité de rechercher dans les documents.

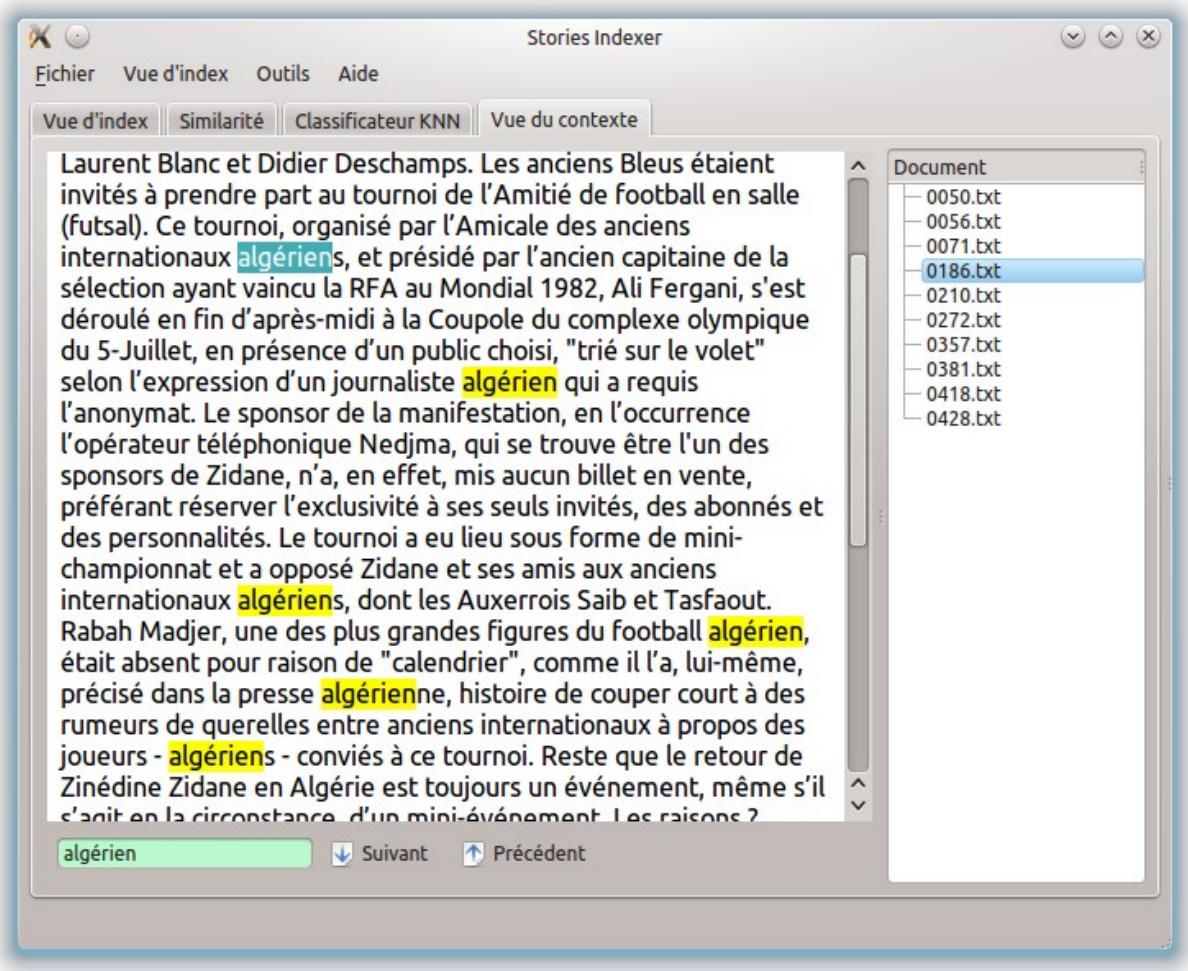

Figure 16: Vue du contexte.

5.4. Similarité

Elle permet de calculer la similarité entre les document indexés et une requête avec les différentes méthodes, on visualise le résultat avec un pourcentage pour chaque document.

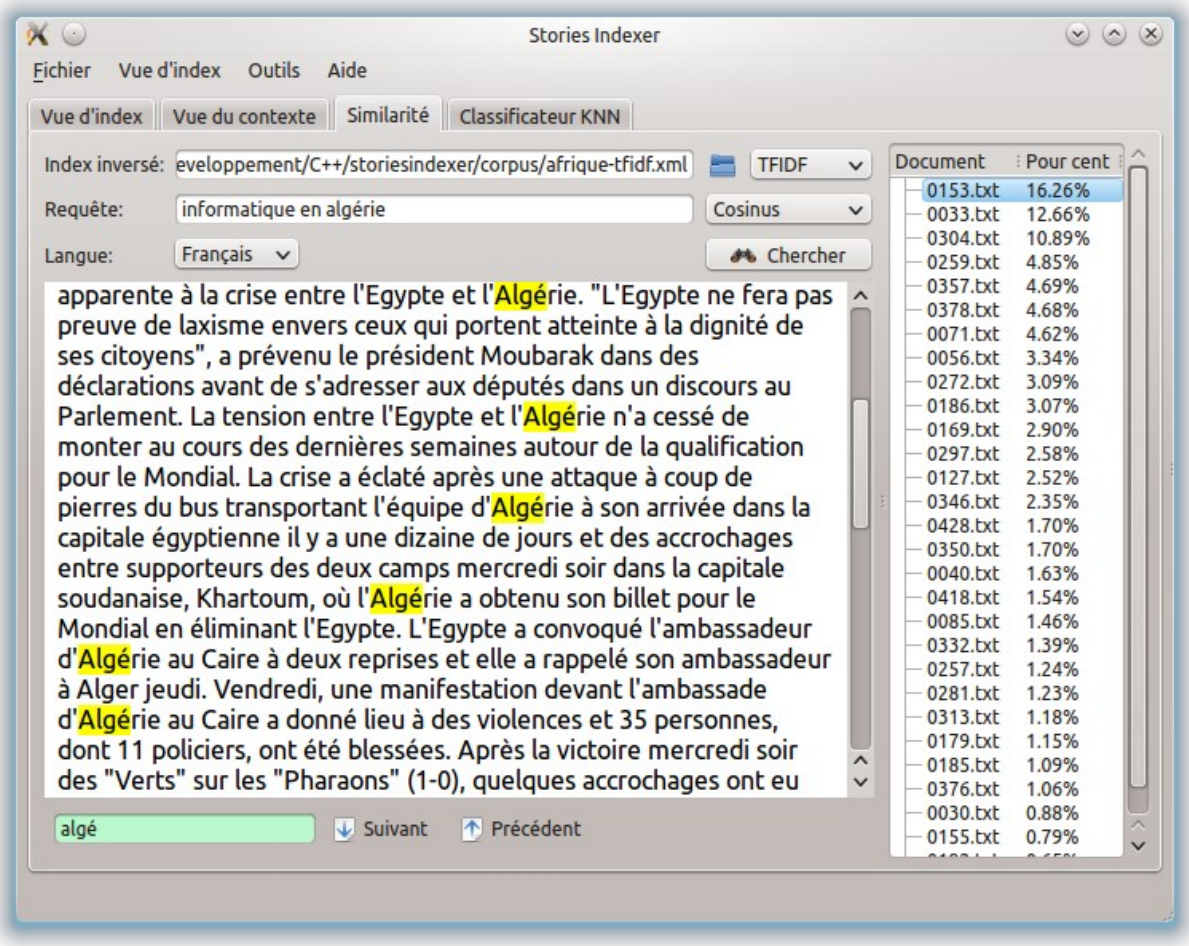

Figure 17: Onglet de similarité.

5.5. Classificateur KNN

Il permet de classifier un nouveau document à l'aide de l'algorithme *kPPV*. On commence par l'étape d'apprentissage d'un corpus d'entraînement. La classification d'un nouveau document se fait par le calcul de similarité.

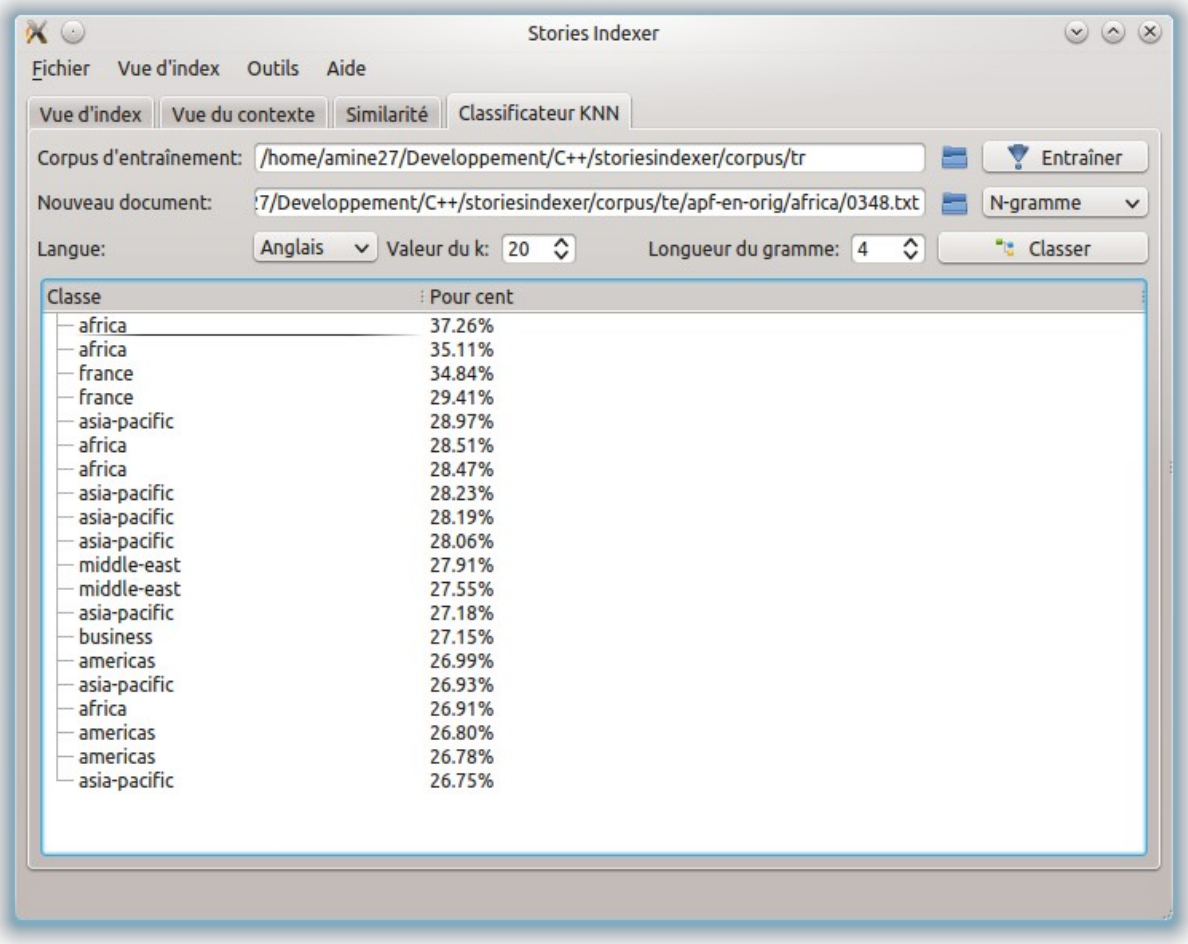

Figure 18: Onglet de classificateur kPPV.

5.6. Configuration

Une interface de configuration de l'application est disponible, elle contient trois onglets :

1. **Général :** Permet de choisir les paramètres par défaut de l'application : la langue d'indexation, la méthode d'indexation, la méthode de similarité et la valeur de K dans l'algorithme kPPV, charger le dernier index au démarrage, demander d'enregistrer l'index avant de fermer et l'affichage permanent des onglets de contexte, similarité et classification.
2. **Apparence :** Changer les apparences de l'application tel que la police à utiliser dans l'afficheur des articles, et la langue de l'application (Français ou l'Anglais).
3. **Indexation :** Paramètres d'indexation : Nombre de top de mots dans chaque documents pour TFIDF et LSI, les k valeurs singulières à garder dans LSI : automatiquement en calculant la racine de *M* (nombre de colonnes) de la matrice LSI ou manuellement. La

matrice initiale de LSI peut être TF, TF normalisé ou TFIDF, activation de l'étape de stemmatisation des termes. Les paramètres de Treetagger sont le chemin de Treetagger, le maximum de documents à tagger dans chaque itération de l'algorithme et l'enregistrement de la sortie de treetagger (documents taggés) dans le répertoire corpus-ttg.

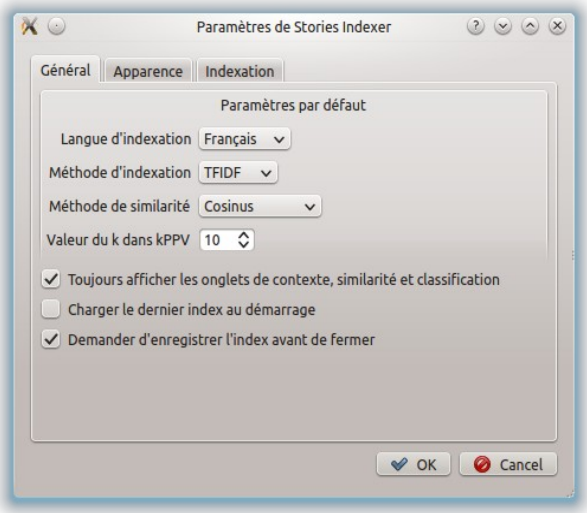

Figure 20: Configuration général.

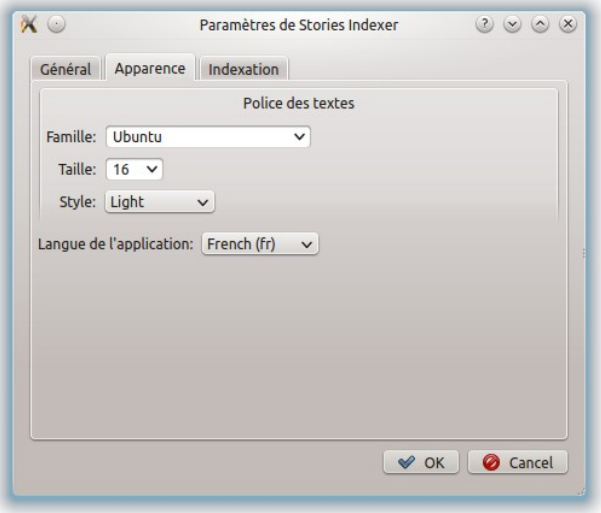

Figure 19: Configuration d'apparence.

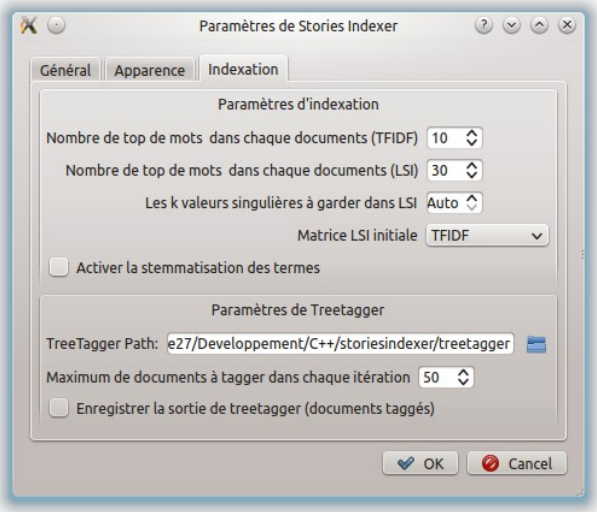

Figure 21: Configuration d'indexation.

5.7. Menu fichier

Il permet d'ouvrir un index, de sauvegarder l'index ou l'index inversé déjà indexé, d'exporter l'index sous format HTML ou PDF.

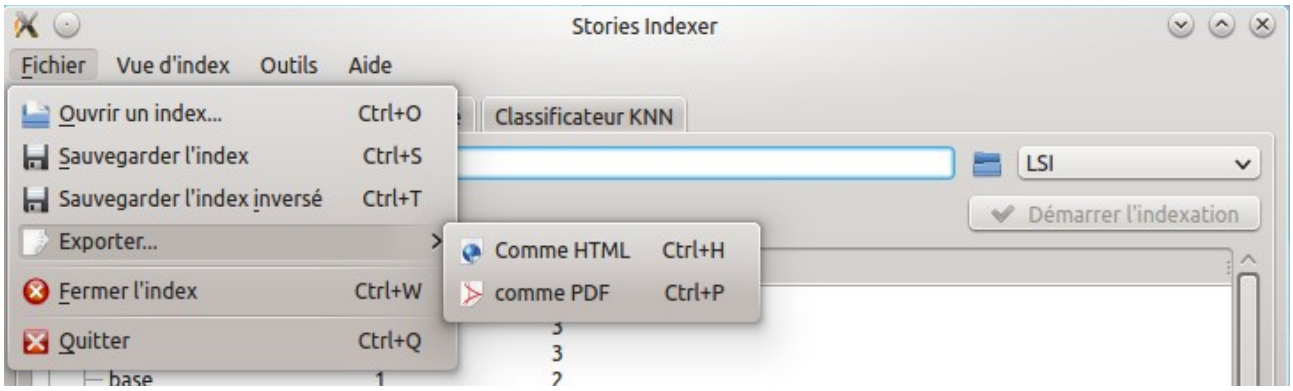

Figure 22: Menu fichier.

5.8. Menu vue d'index

Il permet d'ouvrir les documents qui contiennent le terme sélectionné dans l'index ou de rechercher dans l'index à l'aide de la barre de filtre.

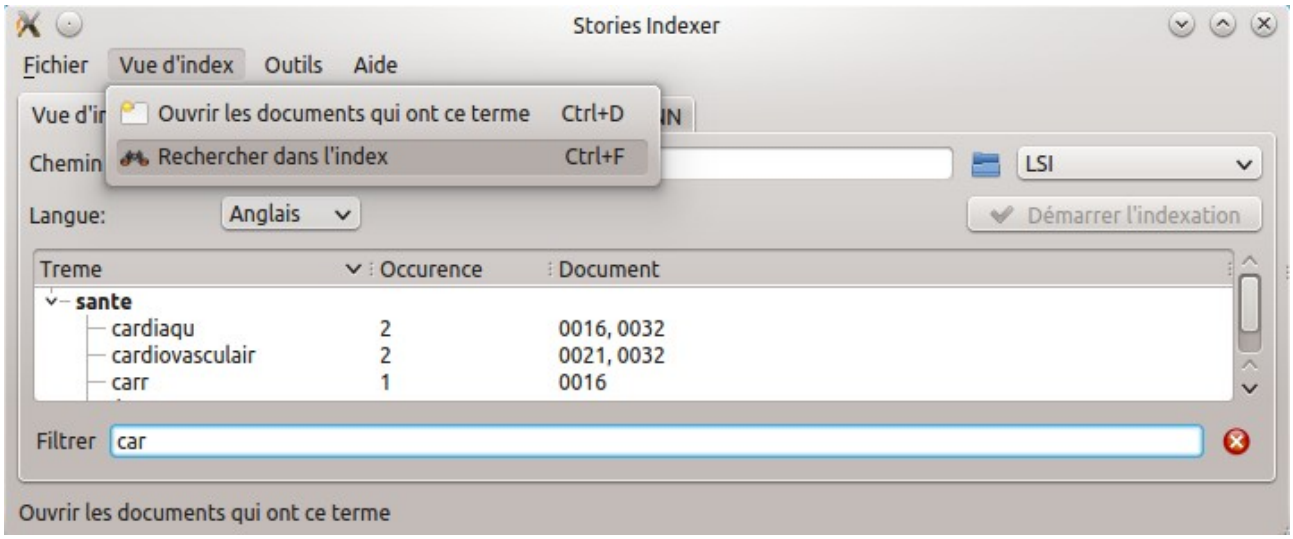

Figure 23: Menu vue d'index.

5.9. Menu outils

Il permet d'ouvrir les onglets de similarité et de classification et le dialogue de paramètres.

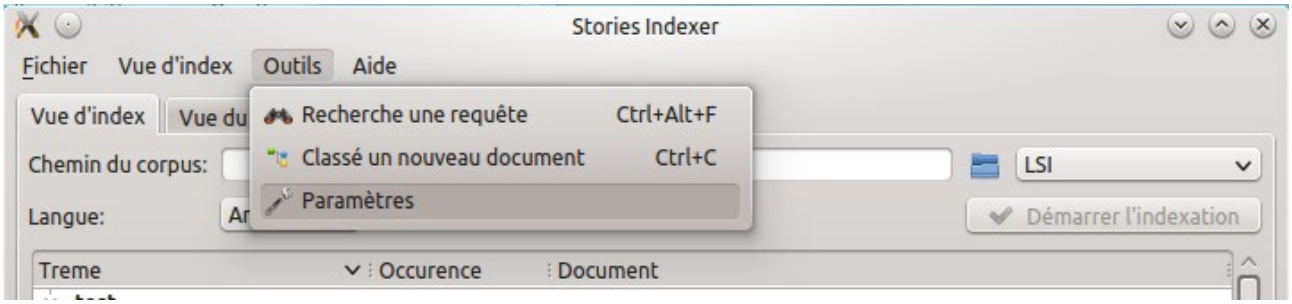

Figure 24: Menu outils.

6. Résultats

6.1. Méthode d'indexation

L'indexation avec la méthode TFIDF fournis des termes significatifs de qualité très bonne dans chaque document et permet de donner une vue globale sur le document. La méthode LSI essaye de trouver les termes synonymes entre les documents, les résultats nous avons vérifié sont de qualité, nous citons comme exemple le terme « stratégie » qui est considéré comme un mot clé pour le document « *Danone réaffirme sa stratégie sur la "santé par l'alimentation"* » (santé-0001.txt) et le document « *Les compléments alimentaires surfent sur la vague de la nutrition santé* » (santé-0004.txt) à cause du terme commun « alimentation ».

6.2. Similarité

La méthodologie d'évaluation des systèmes d'indexation classique depuis [Salton88] définit les notions de taux de rappel et de précision :

$$Rappel = \frac{nombre\ de\ documents\ pertinents\ retrouvés}{nombre\ de\ documents\ retrouvés}$$

Formule 16: Rappel

$$Précision = \frac{nombre\ de\ documents\ pertinents\ retrouvés}{nombre\ de\ documents\ pertinents}$$

Formule 17: Précision

Les mesures de similarité fournissent pour une liste de 5 requêtes (« accidents du travail » « économie de la Grèce » « la coupe d'afrique » « darfour guerre » « lutter contre le sida ») des listes de documents les plus proches. Pour chaque liste et pour chaque document pertinent trouvé

par ordre de similarité décroissante, nous calculons les deux indices, nous comptabilisons ensuite leur moyenne. Ces valeurs sont reportées sur la figure , les courbes rappel / précision respectives.

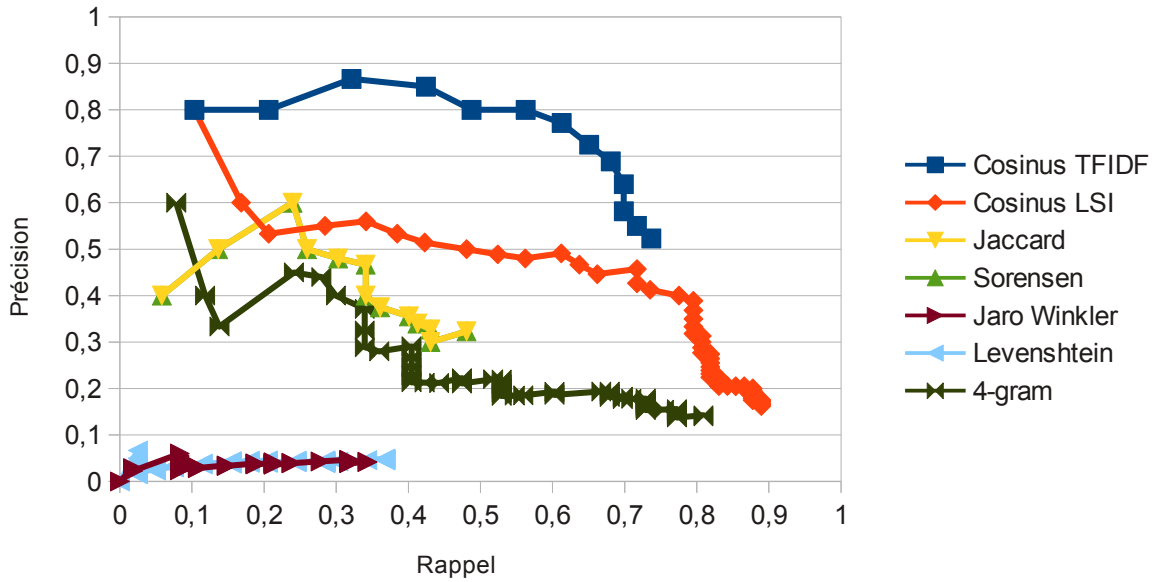

Figure 25: Courbe Rappel/Précision des méthodes de similarité

La méthode de Cosinus TFIDF est largement meilleure que les autres méthodes d'un rappel de de 10% jusqu'à 75%. Ensuite la méthode Cosinus LSI qui donne un rappel jusqu'à 90% qui est le meilleur mais une avec faible précision, les méthodes Jaccard et Sorensen donnent les mêmes résultats puisque les algorithmes sont similaires avec une précision maximale de 60%. L'algorithme de N-Gram appliqué avec N=4 donne aussi une précision moyenne pour un rappel de 7% jusqu'à 14%. Les méthodes Jaro Winkler et Levenshtein donnent des résultats très faibles car elles se basent sur des techniques de substitution et de comparaison de suffix/préfix pour un temps de calcul très supérieur et une grande place mémoire.

Les conclusions immédiates à tirer sont que les méthodes qui sont basées sur la comparaison des chaînes de caractères ne fournissent pas une mesure suffisamment efficace dans le cadre de l'indexation, elles sont largement inférieure à la mesure Cosinus qui utilise une matrice de pondération. Cette expérience montre que d'une manière générale l'exploitation de la notion de pondération apporte une amélioration pour la recherche dans le cadre d'un besoin d'informations.

6.3. Classification

Pour spécifier la meilleure méthode de distance à utiliser dans l'algorithme de kPPV, nous nous sommes intéressé à la mesure du taux de bonne classification simplifié. Il s'agit de l'indicateur le plus naturel et le plus évident permettant d'évaluer les performances d'un système de classification. Cette valeur, simple à calculer, correspond au nombre d'éléments correctement identifiés par le système. La définition du taux de bonne classification est :

$$taux = \frac{Nombre\ d'\ éléments\ correctement\ identifiés}{Nombre\ d'\ éléments\ total}$$

Formule 18: Taux de bonne classification simplifié.

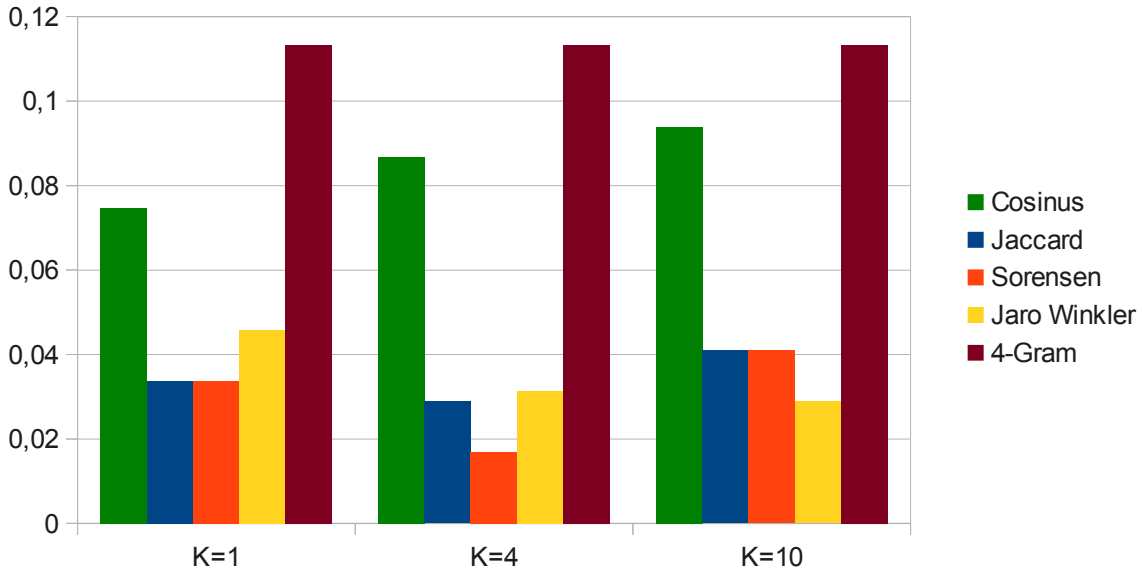

Figure 26: Diagramme du taux de bonne classification.

La distance n-gramme donne les meilleurs résultats, car elle capture automatiquement la racine des mots les plus fréquents. Cette méthode est indépendante de la langue employée dans le corpus et tolérant aux fautes d'orthographe. La méthode cosinus donne aussi des résultats acceptables en augmentant le paramètre k de kppv.

Nous pouvons conclure que les performances de la classification varient significativement d'un document à un autre. Une des raisons de ces différences est la nature du corpus utilisé.

7. Conclusion

Dans ce chapitre nous avons définit le prototype que nous avons implémenté et qui nous a servi pour les expérimentations. Nous avons utilisé le corpus [Roukh10] pour les expérimentations. Ce corpus possède l'avantage d'être une ressource assez fournie disponible en français et en anglais et de disposer des catégories pour chaque documents. Ce système reste néanmoins générique et peut prendre en compte n'importe quel corpus de texte.

Les résultats des méthodes d'indexation fournis des termes significatifs de qualité très bonne dans chaque document et permet de donner une vue globale sur le document

Nos expérimentations portent sur les mesures de similarité proposées dans ce mémoire. Les calculs ont démontré que la méthode de référence Cosinus donne les meilleurs résultats.

Les expérimentations présentent des résultats encourageants. Ils peuvent être améliorés, notamment dans le cadre d'expérimentations sur des corpus de spécialité.

Conclusion générale

Les travaux présentés dans ce mémoire se situent dans le contexte de la recherche d'information, et plus particulièrement dans le cadre de l'indexation des documents numérique.

Un processus d'indexation doit concilier le contenu d'un document pour pouvoir répondre efficacement aux besoins en information des utilisateurs. L'unité retournée par le système de recherche d'information ne doit plus être le document en entier mais des fragments de celui-ci.

Pour ce faire, des solutions concernant le stockage des documents, leur interrogation ainsi que le tri des unités d'information résultats doivent être proposées. Nous nous sommes intéressés dans ce mémoire à proposer une solution flexible pour répondre à de telles problématiques.

Pour valider notre travail, un prototype a été implémenté et nous avons effectué une série d'expérimentations sur des collections de corpus, les résultats sont encourageants. Ils peuvent être améliorés, notamment dans le cadre d'expérimentations sur des corpus de spécialité.

On voit que la tâche est ambitieuse et les difficultés sont nombreuses, mais les pistes que nous avons soulevées dans ce travail nous semblent des objectifs très intéressants à poursuivre.

Abréviations

CDROM : Compact Disc Read Only Memory
CELEX : Communitatis Europeae Lex
CNRS : Centre national de la recherche scientifique
ECODOC : Documentation automatisée en économie générale
IBM : International Business Machines
LSI : Latent Semantic Indexing
PIAF-DOC : Programme Interactif d'Analyse du Français
PRECIS : Preserved Context Indexing System (système d'indexation respectant le contexte)
SINTEX : Système d'INdexation de TEXtes
SPIRIT : Système Syntaxique et Probabiliste d'Interrogation et de Recherche de l'Information Textuelle
SRI : Système de Rcherche d'Information
SYDO : Systèmes Documentaires
SYNTOL : Syntagmatic Organization Language
TF : Term Frequency
IDF : Inverted Document Frequency
UNESCO : United Nations Educational, Scientific and Cultural Organization

Références

[Abbaci03] Faïza Abbaci, Méthodes de sélection de collections dans un environnement de Recherche d'Information distribuée, Université Jean Monnet de St-Etienne, 2003.
[Bannour11] Sondes Bannour, Laurent Audibert, Adeline Nazarenko , Mesures de similarité distributionnelle entre termes. Université Paris 13, 2011.
[Bechet08] N. Bechet, I. Bayoudh , Quelles connaissances linguistiques permettent d'améliorer la classification de blogs avec les k-ppv ?, Quatrième Atelier : Qualité des Données et des Connaissances, 2008.
[Bensiali07] Bensiali A, Indexation de documents semi-structurés «Proposition d'une amélioration de l'index IFTI». Thèse d'Ingénieur de l'Institut National de formation en Informatique (I.N.I) Alger, 2007.
[Berrut97] Berrut C, Indexation des données multimédia, utilisation dans le cadre d'un système de recherche d'informations. Thèse pour obtenir le diplôme d'Habilitation à Diriger des Recherches. Université Joseph Fourier - Grenoble I, 1997.
[Bianco05] Bianco, M., Dessus, P., et.al, Modélisation des processus de hiérarchisation et d'application de macrorègles et conception d'un prototype d'aide au résumé. In G. Denhière (Ed.), Le résumé de textes : de l'Analyse Sémantique Latente à l'élaboration d'un tuteur électronique. Paris, 2005.
[Dahak06] Dahak F, Indexation des documents Semi-Structurés : Proposition d'une approche basée sur le fichier inversé et le "Trie". Thèse de Magister de l'Institut National de formation en Informatique (I.N.I) Alger, 2006.
[Deerwester90] Deerwester S., Dumais S. T., Furnas G. W., Landauer T. K., Harshman R., Indexing by latent semantic analysis. Journal of the American Society for Information Science, 1990.
[Deerwester90] Scott Deerwester, Susan Dumais, et.al, Indexing by Latent Semantic Analysis, dans Journal of the Society for Information Science, 1990.
[Denoyer04] Denoyer L, Apprentissage et inférence statistique dans les bases de documents structurés : Application aux corpus de documents textuels. Thèse de doctorat de l'université de PARIS 6, 2004.
[Dunning94] Dunning Ted, Statistical identification of language. Computing Research Lab, New Mexico State University, 1994.
[Dupas10] Eve Mathieu-Dupas, Algorithme des K plus proches voisins pondérés (WKNN) et Application en diagnostic. 42èmes Journées de Statistique, 2010.
[Dziczkowski08] G. Dziczkowski , Analyse des sentiments : système autonome d'exploration des opinions exprimées dans les critiques cinématographiques. Thèse de doctorat, Ecoles Supérieures des Mines de Paris, 2008.
[Hachani97] Mabrouka el Hachani, L'indexation automatique. Ecole Nationale Supérieure des Sciences de l'Information et des Bibliothèques, 1997.
[Harbeck99] Harbeck S. et Ohler U., Multigrams for Language Identification. Proceedings of the 6th European Conference on Speech Communication and Technology, Budapest, Hungary, 1999.
[Hudon98] Michèle Hudon, Indexation et langages documentaires dans les milieux archivistiques à l'ère des nouvelles technologies de l'information, 1998.
[Jaccard1901] Paul Jaccard, Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bulletin de la Société Vaudoise des Sciences Naturelles, 1901.
[Jacquemin00] Jacquemin C. et Zweigenbaum P., Traitement automatique des langues pour l'accès au contenu des documents.Le document Multimédia en Sciences du Traitement de l'Information, CÉPADUÈS-Éditions, Toulouse, 2000.
[Jaillet03] Jaillet S, Teisseire M, Dray G, Adéquation des modèles de représentation aux méthodes de catégorisation. LIRMM-CNRS -ISIM - Université Montpellier, 2003.

[Jaillet03] Jaillet S., Chauché J., Prince V., Teisseire M. Classification automatique de documents: la mesure de deux écarts. Rapport de Recherche LIRMM 18p, 2003.
[Kontostathis06] April Kontostathis, William M. Pottenger, A framework for understanding Latent Semantic Indexing (LSI) performance. Inf. Process. Manage, 2006.
[Lancaster98] Lancaster F.-W, Indexing and abstracting in theory and practice. Library Association Publishing. London, 1998.
[Landauer97] Landauer, T. K., & Dumais, S. T., A solution to Plato's problem : the Latent Semantic Analysis theory of acquisition, induction and representation of knowledge. Psychological Review, 1997.
[Laporte00] Laporte Eric, Mots et niveau lexical. Ingénierie des langues, Hermes, 2000.
[Laroum09] Laroum S., Béchet N., Hamza H., Roche M., Classification automatique de documents bruités à faible contenu textuel. Revue des Nouvelles Technologies de l'Information, 2009.
[Levenshtein66] Levenshtein VI, Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10, 1966.
[Lévesque02] M. Lévesque, L'indexation: luxe ou nécessité?. Archives , Volume 33, Numéro1, 2001-2002.
[Lien01] http://snowball.tartarus.org/algorithms/french/stop.txt 2012.
[Lien02] http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop 2012.
[Luhn58] H.P. Luhn, The automatic creation of literature abstracts. Journal of Research and Development d'IBM, 1958.
[Moreau06] Moreau F, Revisiter le couplage traitement automatique des langues et recherche d'information. Thèse de doctorat de l'Université de Rennes 1, 2006.
[Paradis96] Paradis F, Un modèle d'indexation pour les documents textuels structurés. Thèse de doctorat de l'Université Joseph Fourier - Grenoble 1, 1996.
[Piwowarski03] B. Piwowarski, Techniques d'apprentissage pour le traitement d'informations structurées: application à la recherche d'information. Thèse pour l'obtention du grade de Docteur de l'Université Paris 6, 2003.
[Porter80] M.F. Porter, An algorithm for suffix stripping. Program 14 no. 3, pp 130-137, July 1980.
[Pouliquen02] Pouliquen B, Indexation de textes médicaux par extraction de concepts, et ses utilisations. Thèse de doctorat de l'université de Rennes I, 2002.
[Rijsbergen80] V.Rijsbergen C. J, Information Retrieval. Department of Computing Science University of Glasgow, 1980.
[Roche06] Mathieu Roche, Jacques Chauché, LSA : les limites d'une approche statistique. Équipe TAL, LIRMM - UMR 5506, Université Montpellier 2, 2006.
[Roukh10] A. Roukh, A Sadouki, Réalisation d'un Crawler pour la recherche d'information Arabe sur le Web. Université de Mostaganem, 2010.
[Roussey01] Roussey C, Une méthode d'indexation sémantique adaptée aux corpus multilingues. Thèse de doctorat, L'Institut National des Sciences Appliquées de Lyon, 2001.
[Salton88] Salton, G. et C. Buckley, Term-weighting approaches in automatic text retrieval. Information Processing & Management, 1988.
[Sauvagnat05] Sauvagnat K, Modèle flexible pour la Recherche d'Information dans des corpus de documents semi-structurés. Thèse en vue de l'obtention du Doctorat de l'Université Paul Sabatier, 2005.
[Schmid94] Helmut Schmid, Probabilistic Part-of-Speech Tagging Using Decision Trees. Proceedings of International Conference on New Methods in Language Processing, Manchester, UK, 1994.
[Sidhom02] Sidhom S, Plate-forme d'analyse morpho-syntaxique pour l'indexation automatique et la recherche d'information : de l'écrit vers la gestion des connaissances. Thèse de doctorat de l'université de Claude Bernard – Lyon 1, 2002.
[Sørensen48] Sörensen T. A method of establishing groups of equal amplitude in plant sociology based on similarity of species content. Kongelige Danske Videnskabernes Selskab, 1948.
[Tebri04] H. Tebri, Formalisation et spécification d'un système de filtrage incrémental

d'information. Thèse en vue de l'obtention du Doctorat de l'Université Paul Sabatier, 2004.
[Weigel02] F. Weigel, A Survey of Indexing Techniques for Semistructured Documents. Projektarbeit im Rahmen des Fortgeschrittenenpraktikums, Ludwig Maximilians Universität München, 2002.
[Winkler99] Winkler, W. E., The state of record linkage and current research problems. Dans Statistics of Income Division, Internal Revenue Service Publication, 1999.
[Zargayouna05] H. Zargayouna, Indexation sémantique de documents XML. Thèse présentée pour l'obtention du grade de Docteur en sciences de l'université Paris XI Orsay, 2005.

Annexe 1 : SVD

L'algorithme prend comme paramètre la matrice terme-document pondérée X et effectue une décomposition en valeurs singulières, qui donne deux matrices orthonormales U et V et une matrice diagonale Σ. On a alors :

$$X = U \Sigma V^T$$

Les produits de matrice qui donnent les corrélations entre les termes d'une part et entre les documents d'autre part s'écrivent alors :

$$XX^T = (U \Sigma V^T)(U \Sigma V^T)^T = (U \Sigma V^T)(V^{TT} \Sigma^T U^T) = U \Sigma V^T V \Sigma^T U^T = U \Sigma \Sigma^T U^T$$
$$X^T X = (U \Sigma V^T)^T (U \Sigma V^T) = (V^{TT} \Sigma^T U^T)(U \Sigma V^T) = V \Sigma^T U^T U \Sigma V^T = V \Sigma^T \Sigma V^T$$

Puisque les matrices $\Sigma \Sigma^T$ et $\Sigma^T \Sigma$ sont diagonales, U est faite des vecteurs propres de XX^T, et V est faite des vecteurs propres de . $X^T X$ Les deux produits ont alors les mêmes valeurs propres non-nulles, qui correspondent aux coefficients diagonaux non-nuls de $\Sigma \Sigma^T$. La décomposition s'écrit alors :

$$\begin{array}{ccccccc} & & X & & U & \Sigma & V^T \\ & & (\mathbf{d}_j) & & & & (\hat{\mathbf{d}}_j) \\ & & \downarrow & & & & \downarrow \\ (\mathbf{t}_i^T) \rightarrow & \begin{pmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{pmatrix} & = & (\hat{\mathbf{t}}_i^T) \rightarrow & \begin{pmatrix} \begin{pmatrix} \\ \mathbf{u}_1 \\ \\ \end{pmatrix} \dots \begin{pmatrix} \\ \mathbf{u}_l \\ \\ \end{pmatrix} \end{pmatrix} & \cdot \begin{pmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{pmatrix} & \cdot \begin{pmatrix} (& \mathbf{v}_1 &) \\ & \vdots & \\ (& \mathbf{v}_l &) \end{pmatrix} \end{array}$$

Les valeurs σ_1,..... σ_l sont les valeurs singulières de X. D'autre part, les vecteurs μ_1,..., μ_l et ν_1,..., ν_l sont respectivement singuliers à gauches et à droite.

On remarque également que la seule partie de U qui contribue à t_i est la $i^{\text{ième}}$ ligne. On note désormais ce vecteur $\hat{t}_i$. De même la seule partie de V^T qui contribue à d_j est la $j^{\text{ième}}$ colonne, que l'on note $\hat{d}_j$.

Annexe 2 : Algorithmes de classifications

Machine à vecteur de support

Les machines à vecteur de support (SVM) sont à l'origine de nouvelles méthodes de catégorisations [Jaillet03], le principe des SVM consiste en une stratégie de minimisation structurelle du risque. En ce qui concerne son application à la problématique de catégorisation de documents, l'approche par SVM permet de définir, par apprentissage, une surface de séparation

entre des exemples positifs et négatifs minimisant le risque d'erreur et maximisant la marge entre deux classes. La **Figure 27** montre une telle séparation dans le cas d'une séparation linéaire par un hyperplan. Il est intéressant de remarquer qu'en réduisant le jeu d'entraînement uniquement aux vecteurs de support, l'algorithme calculerait le même hyperplan que pour le jeu d'entraînement complet. La marge se présente alors comme la plus courte distance entre un vecteur de support et "son" hyperplan.

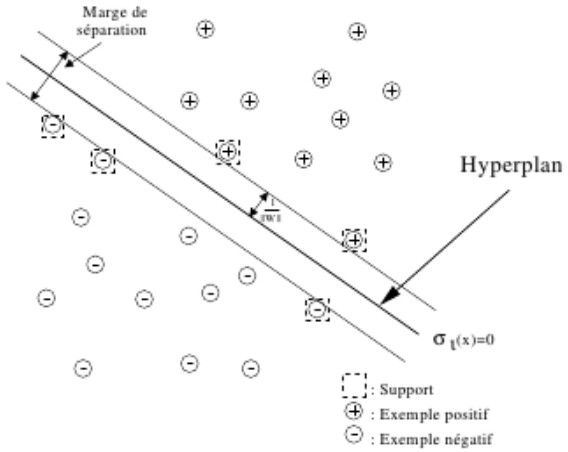

Figure 27: Machine à vecteur de support

De manière formelle, un hyperplan peut être défini par :

$$\vec{w}\cdot\vec{x}+b=0$$

Avec x un point arbitraire, w un vecteur et b le biais.

Soit D={(x_i, y_i)} le jeu d'entraînement et $y_i \in \{\pm 1\}$ définissant l'état, positif ou négatif, de l'exemple. Trouver l'hyperplan maximisant la marge séparatrice revient à résoudre le problème suivant :

$$\begin{cases} minimiser \dfrac{1}{2}\|w\|^2 \\ sous\ les\ contraintes\ \forall i,\ y_i(\vec{w}\cdot\vec{x}_i+b)-1\geq 0 \end{cases}$$

Grâce à une extension de cet algorithme, il est aussi possible de résoudre des problèmes qui ne sont pas linéairement séparables, mais l'amélioration obtenue pour la catégorisation de documents reste minime. Pour la construction vectorielle des textes, ce sont en général les stems (radicaux) qui sont utilisés comme termes d'indexation [Jaillet03].

SVM est considéré comme un des algorithmes les plus performants en classification textuelle [Laroum09].

Naïve Bayes

Le classificateur Naïve Bayes est un catégoriseur de type probabiliste fondé sur le théorème de Bayes (1763). Considérons $v_j = (v_{j1}, ..., v_{jk}, ..., v_{jd})$ un vecteur de variables aléatoires représentant un document d_j et C un ensemble de classes.

En s'appuyant sur le théorème de Bayes, la probabilité que ce dernier appartienne à la classes $c_i \in C$ est définie par [Laroum09]:

$$P(c_i|v_j) = \frac{P(c_i)P(v_j|c_j)}{P(v_j)}$$

Formule 19: Théorème de Bayes.

Ce théorème repose sur l'hypothèse que des solutions recherchées peuvent être trouvées à partir de distributions de probabilité dans les hypothèses et dans les données.

Un classificateur bayésien naïf, dans le cadre de la classification de textes, permet de déterminer la classe d'un document spécifié en supposant que les documents sont indépendants. Cette hypothèse d'indépendance ne reflète pas la réalité d'où l'appellation naïf. La classe la plus probable d'un nouvel objet est déterminée en combinant les prédictions de toutes les hypothèses en les pondérant par leurs probabilités a priori [Bechet08].

La variable aléatoire v_{jk} du vecteur v_j représente l'occurrence de l'unité linguistique k retenue pour la classification dans le document d_j. La classe c_k d'appartenance de la représentation vectorielle v_j d'un document d_j est définie comme suit:

$$c_k = arg\ max\ P(c_i \in C) \prod_k P(v_{jk}|c_j)$$

En d'autres termes, le classificateur Naïve Bayes affecte au document d_j la classe ayant obtenue la probabilité d'appartenance la plus élevée. Alors, $P(c_i)$ est définie de la façon suivante :

$$P(c_i) = \frac{nombre\ de\ documents \in c_i}{nombre\ total\ de\ documents}$$

En faisant l'hypothèse que les v_j sont indépendantes, la probabilité conditionnelle $P(v_j|c_i)$ est définie ainsi:

$$P(v_j|c_i) = P(v_{jk}|c_i)$$

Ce classificateur s'est montré moins performant pour des tâches de classification de textes que d'autres méthodes [Bechet08].

Arbres de décision

L'arbre de décision est un arbre de nœuds internes qui sont marqués par des termes, les branches qui sortent des nœuds sont des tests sur les termes, et les feuilles sont marquées par catégories. Ce classificateur classe un document du test d_j en testant récursivement les poids des nœuds internes de vecteur d_j , jusqu'à ce qu'une feuille soit atteinte. L'étiquette de ce nœud est alors attribuée à d_j. La plupart de ces classificateurs utilise une représentation du document binaire, et sont donc crée par des arbres binaires.

Une méthode pour effectuer l'apprentissage d'un arbre de décision pour la catégorie c_i consiste à vérifier si tous les exemples d'apprentissage ont la même étiquette (c_i), dans le cas contraire nous sélectionnons un terme t_k, et nous partitionnons l'ensemble d'apprentissage en classes de documents qui ont la même valeur pour t_k , et à la fin l'on crée les sous-arbres pour chacune de ces classes. Ce processus est répété récursivement sur les sous-arbres jusqu'à ce que chaque feuille de l'arbre généré de cette façon contienne des exemples d'apprentissage attribués à la même catégorie c_i , qui est alors choisie comme l'étiquette de la feuille. L'étape la plus importante est le choix du terme de t_k pour effectuer la partition [Dziczkowski08].

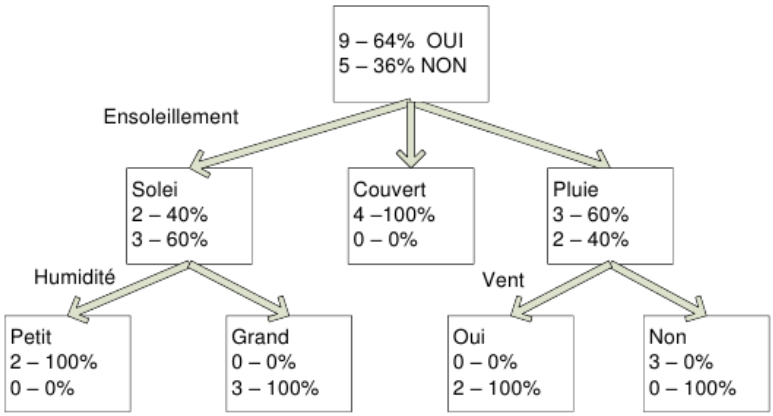

Figure 28: Exemple d'un arbre de décision.

Un exemple d'un arbre de décision est montrée sur la **Figure 28**. Il s'agit de prédire le comportement de sportifs (Jouer ; variable à prédire) en fonction de données météo (Ensoleillement, Température, Humidité, Vent ; variables prédictives). Sur chaque sommet de l'arbre est décrite la distribution de la variable à prédire. Dans le cas du premier sommet, la racine de l'arbre, nous constatons qu'il y a 14 observations dans notre fichier, 9 d'entre eux ont décidé de jouer (Jouer = oui), 5 ont décidé le contraire (Jouer = non). Ce premier sommet est segmenté à l'aide de la variable Ensoleillement, 3 sous-groupes ont été produits. Le premier groupe à gauche (Ensoleillement = Soleil) comporte 5 observations, 2 d'entre eux correspondent à Jouer = oui, 3 à Jouer = non. Chaque sommet est ainsi itérativement traité jusqu'à ce que l'on obtienne des groupes suffisamment

homogènes. Elles correspondent aux feuilles de l'arbre, des sommets qui ne sont plus segmentés.

Réseau de neurones

Un classificateur de texte basé sur les réseaux de neurones est un réseau d'unités, où les unités d'entrée représentent les termes, l'unité(s) de sortie représentent la catégorie ou les catégories d'intérêts, et le poids sur les bords reliant les unités représentent les relations de dépendance. Pour classer un document de test d_j, ses poids w_{kj} sont chargés dans les unités d'entrée ; l'activation de ces unités se propage à travers le réseau, et la valeur de l'unité de sortie(s) détermine la décision du classement. Une manière typique d'apprentissage de réseau de neurones est la rétro propagation, qui consiste à rétro propager l'erreur commise par un neurone à ses synapses et aux neurones qui y sont reliés. Pour les réseaux de neurones, on utilise habituellement la rétro propagation du gradient de l'erreur, qui consiste à corriger les erreurs selon l'importance des éléments qui ont justement participé à la réalisation de ces erreurs [Dziczkowski08].

En règle générale, le calcul de la valeur de neurone peut se décomposer en deux étapes :

- Une combinaison linéaire des entrées :

$$v = w_0 + \sum_{i=1}^{n} (w_i \cdot x_i)$$

- La sortie du neurone est :

$$y = f(v) = f\left(\sum_{i=0}^{n} (w_i \cdot x_i)\right)$$

w_0 est le biais, il peut être considéré comme la pondération de l'entrée 0 fixée à 1.

v est le potentiel du neurone.

La fonction f est la fonction d'activation du neurone.

Un classificateur linéaire est un type de classificateur le plus simple. Dans cet algorithme, le classificateur de c_i est d'abord initialisé par la mise de tous les poids w_{kj} à la même valeur positive. Quand un exemple d'apprentissage d_j est examiné, si le résultat de la classification est correct, rien n'est fait, alors que si le résultat est faux, les poids du classificateur sont modifiés :

- si d_j est un exemple positif de c_i, le poids de w_{ki} des termes actifs (c'est-à-dire, les termes t_k tels que $w_{kj} = 1$) sont "promus" par augmentation d'une valeur $\alpha > 0$ (appelé taux d'apprentissage),
- si d_j est un exemple négatif de c_i les mêmes poids sont "rétrogradés" en diminuant leur valeur par α.

Lorsque le classificateur a atteint un niveau raisonnable d'efficacité, le fait qu'un poids w_{ki} est très faible signifie que t_k a contribué négativement à la procédure de classement, il peut donc être éliminé de la représentation.

Un réseau de neurones est en général composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente. Chaque couche est composée de n_i neurones, prenant leurs entrées sur les n_{i-1} neurones de la couche précédente. À chaque synapse est associé un poids synaptique, de sorte que les n_{i-1} sont multipliés par ce poids, puis additionnés par les neurones de niveau *i*, ce qui est équivalent à multiplier le vecteur d'entrée par une matrice de transformation. Le réseau de neurones peut également contenir des boucles qui en changent radicalement les possibilités mais aussi la complexité. La représentation schématique d'un neurone artificiel avec un index *j* est montre sur la **Figure 29** [Dziczkowski08].

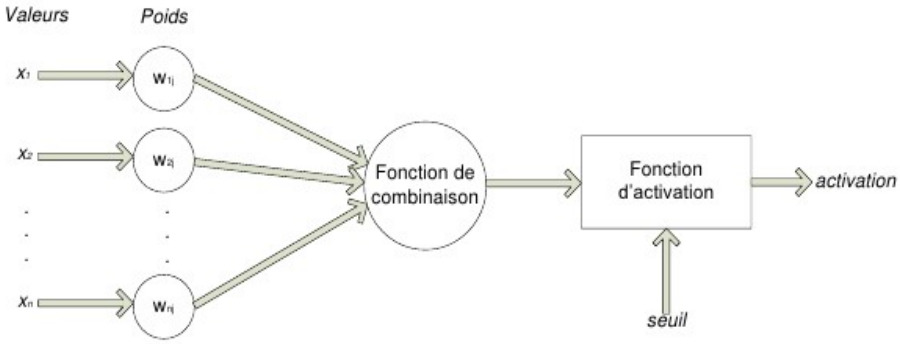

Figure 29: Structure d'un neurone artificiel